Learning I:
Learning to Adjust, Learning to Predict

Purpose/benefits of Learning

- More believable (less predictable) characters

- Less work to program decisions for all the possible situations

Purpose/benefits of Learning

Challenges using learning in games?

- More believable (less predictable) charac

Lack of control over learnt behaviors

- Less work to program decisions f ossible situations

Overfitting

Types of Learning

- Offline
  - -between levels of games
  - -at the game development studio before the game is released

Most common approach

Allows to test the learnt behaviors

- Online
  - -during the game itself

Types of Learning

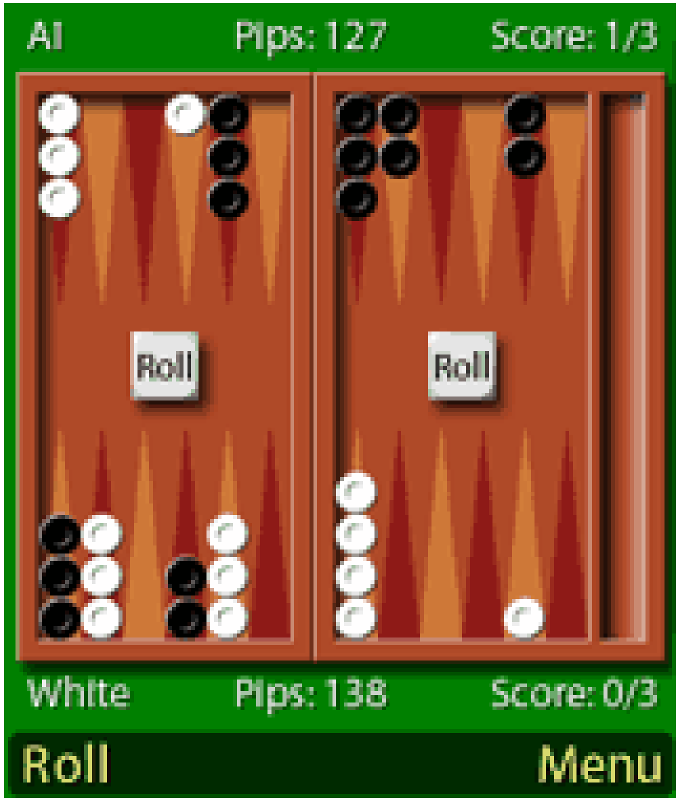
- Adaptation of the behavior parameters (intra-behavior learning)
    - learning to target precisely
    - learning the best patrol routes on the current level
    - learning good set of cover points for the given room
    - …

Types of Learning

- Predicting the behavior of the enemy
    - predicting the next move in a fight based on the last few moves
    - predicting the attack based on the assembly of enemy troops
    - …

Types of Learning

- Reacting to the behavior of the enemy (inter-behavior learning)
  - making the most effective counter-move in a fight
  - making the most effective reallocation of the troops
  - making the best move in a board game based on all the pieces
  - …



Adaptation of the Parameters

- Hill Climbing

The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = \text{argmax}_{\Psi} f(\Psi)$

Start with the best guess for the parameter vector Ψ

Until no further improvement

try changing Ψ *by small* Δ *in all directions and pick the best:*

$$\Psi = \Psi + \text{argmax}_{\Delta} f(\Psi+\Delta)$$

Adaptation of the Parameters

- Hill Climbing

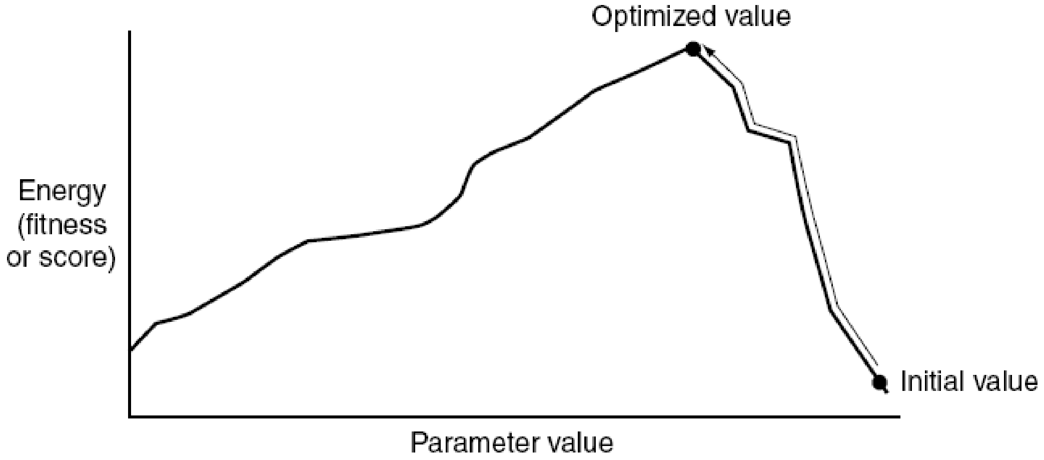
The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = \mathrm{argmax}_{\Psi} f(\Psi)$

Start with the best guess for the parameter vector Ψ

Until no further improvement

try changing Ψ *by small* Δ *in all directions and pick the best:*

$$\Psi = \Psi + \mathrm{argmax}_{\Delta} f(\Psi+\Delta)$$



from "Artificial Intelligence for Games" by I. Millington & J. y ge

Adaptation of the Parameters

- Hill Climbing

The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = \mathrm{argmax}_{\Psi}\, f(\Psi)$

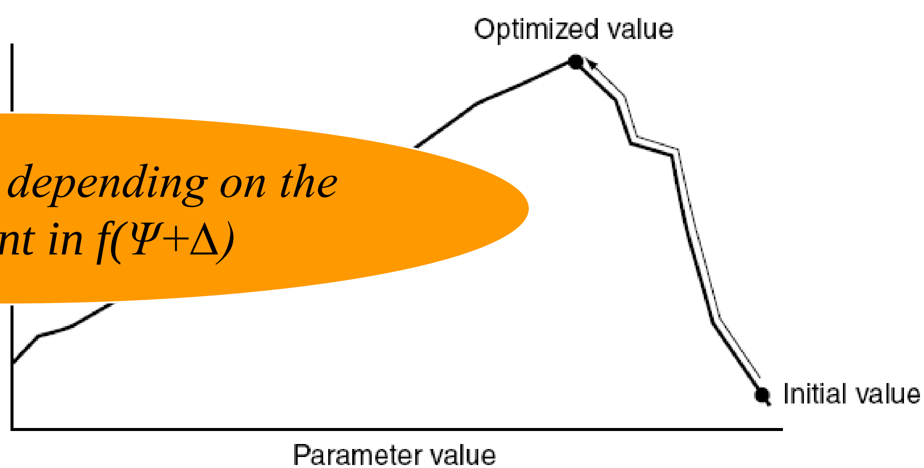
Start with the best guess for the parameter vector Ψ

Until no further improvement

try changing Ψ *by small* Δ *in all directions and pick the best:*

$$\Psi = \Psi + \mathrm{argmax}_{\Delta}\, f(\Psi+\Delta)$$

Δ is often adapted depending on the rate of improvement in $f(\Psi+\Delta)$



from “Artificial Intelligence for Games” by I. Millington & J. y ge

Adaptation of the Parameters

- Hill Climbing

The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = \mathrm{argmax}_{\Psi}\, f(\Psi)$

Start with the best guess for the parameter vector Ψ

Until no further improvement

try changing Ψ *by small* Δ *in all directions and pick the best:*

$$\Psi = \Psi + \mathrm{argmax}_{\Delta}\, f(\Psi+\Delta)$$

Example:

finding the best shooting distance

Ψ *= distance (scalar variable)*

Δ *= +/-distance increment*

$f(\Psi)$ *= distance between enemy ship and where cannonball lands*

Adaptation of the Parameters

- Hill Climbing

The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = argmax_{\Psi} f(\Psi)$

Start with the best guess for the parameter vector Ψ

Until no further improvement

try changing Ψ *by small* Δ *in all directions and pick the best:*

$$\Psi = \Psi + argmax_{\Delta} f(\Psi+\Delta)$$

Equivalent formulation for differentiable functions (Steepest ascent):

Start with the best guess for the parameter vector X

Until no further improvement

$$X = X + \varepsilon f'(X)$$

Adaptation of the Parameters

- Hill Climbing

The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = argmax_{\Psi} f(\Psi)$

Start with the best guess for the parameter vector Ψ

Until no further improvement

try changing Ψ *by small* Δ *in all directions and pick the best:*

$\Psi = \Psi + argmax$ …

$X = X + argmax_{\Delta} f(X+\Delta)$ *for small* $|\Delta|$,

$X \approx X + argmax_{\Delta} \{f(X) + f'(X)\Delta\}$

$X = X + \varepsilon f'(X)$ *for small* $\varepsilon>0$

Why?

Equivalent formulation for diffe… …ent):

Start with the best guess for the parameter vector X

Until no further improvement

$$X = X + \varepsilon f'(X)$$

Adaptation of the Parameters

- Hill Climbing

The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = \mathrm{argmax}_{\Psi} f(\Psi)$

Start with the best guess for the parameter vector Ψ

Until no further improvement

try changing Ψ *by small* Δ *in all directions and pick the best:*

$$\Psi = \Psi + \mathrm{argmax}_{\Delta} f(\Psi+\Delta)$$

Equivalent formulation for differentiable functions (Steepest ascent):

Start with the best guess for the parameter vector X

Until no further improvement

$$X = X + \varepsilon \nabla f(X)$$

For multi-dimensional vector X,

$f'(X) \equiv \nabla f(X)$

Adaptation of the Parameters

- Hill Climbing

The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = \operatorname{argmax}_{\Psi} f(\Psi)$

Start with the best guess for the parameter vector Ψ

Until no further improvement

try changing Ψ *by small* Δ *in all directions and pick the best:*

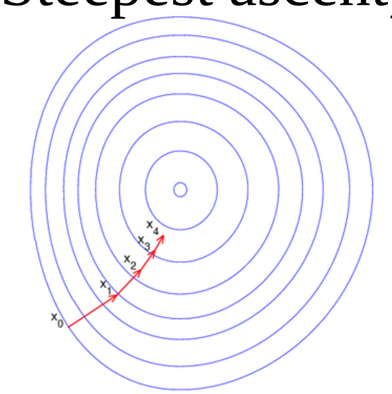
$$\Psi = \Psi + \operatorname{argmax}_{\Delta} f(\Psi+\Delta)$$

Equivalent formulation for differentiable functions (Steepest ascent):

Start with the best guess for the parameter vector X

Until no further improvement

$$X = X + \varepsilon \nabla f(X)$$



Adaptation of the Parameters

- Hill Climbing

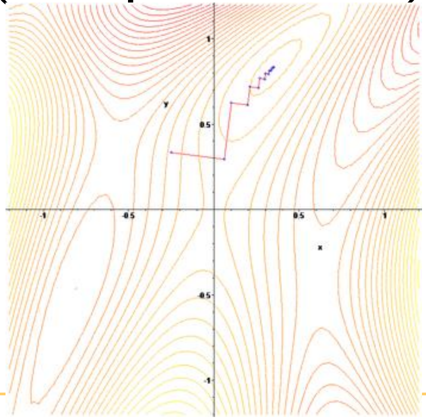
The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = \mathrm{argmax}_{\Psi}\, f(\Psi)$

Start with the best guess for the parameter vector Ψ
Until no further improvement
try changing Ψ *by small* Δ *in all directions and pick the best:*

$$\Psi = \Psi + \mathrm{argmax}_{\Delta}\, f(\Psi+\Delta)$$

Equivalent formulation for differentiable functions (Steepest ascent):
Start with the best guess for the parameter vector X
Until no further improvement

$$X = X + \varepsilon \nabla f(X)$$



Adaptation of the Parameters

- Hill Climbing

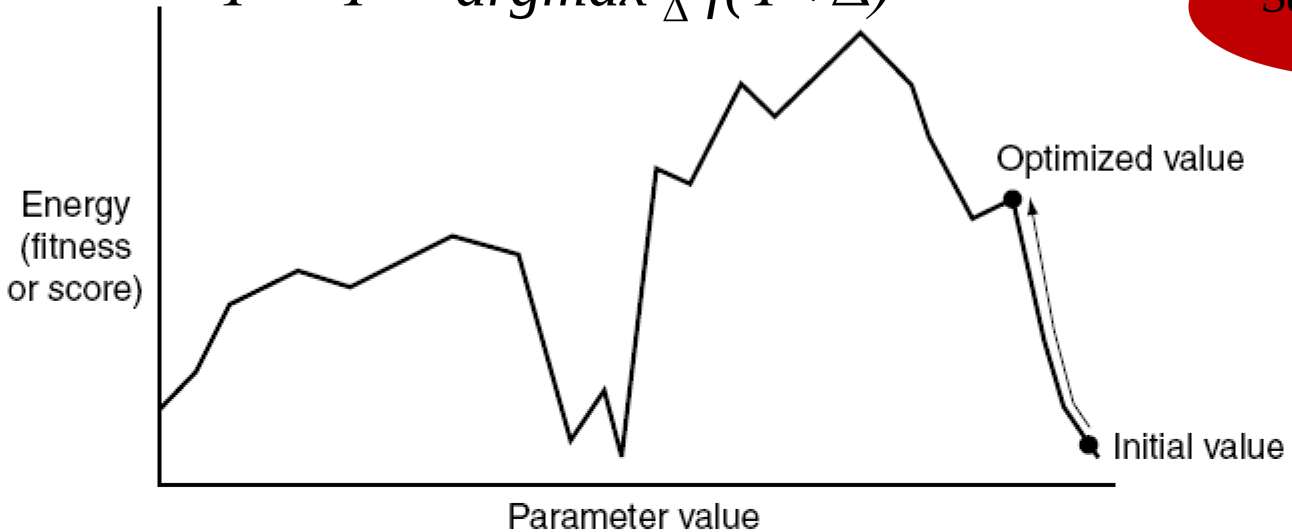
The goal is to find the parameter vector that results in the most optimal value of the function: $\Psi^* = \mathrm{argmax}_{\Psi} f(\Psi)$

Start with the best guess for the parameter vector Ψ
Until no further improvement
try changing Ψ *by small* Δ *in all directions and pick the best:*

$$\Psi = \Psi + \mathrm{argmax}_{\Delta} f(\Psi+\Delta)$$

Typical problems?

Solutions?



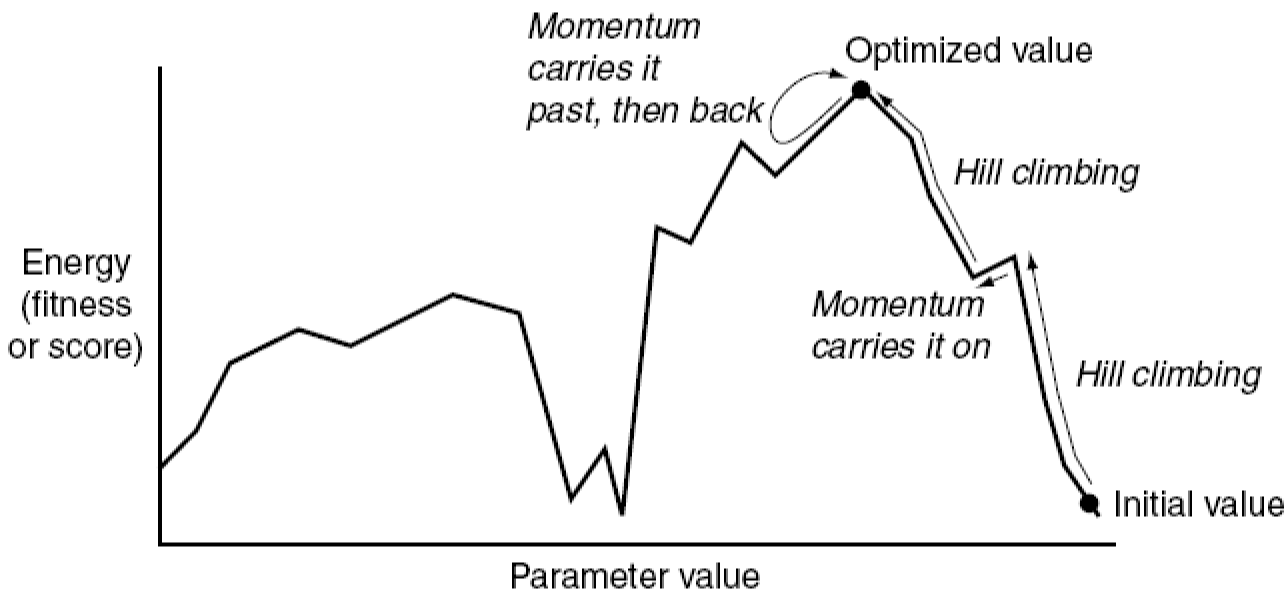
from "Artificial Intelligence for Games" by I. Millington & J. y ge

Adaptation of the Parameters

- Extensions to Hill Climbing to overcome local minima problem

  \- Momentum: persist in going in the same direction (*when picking the next Δ, the previous direction Δ gets an additional term proportional to the previous improvement)*

Disadvantages?



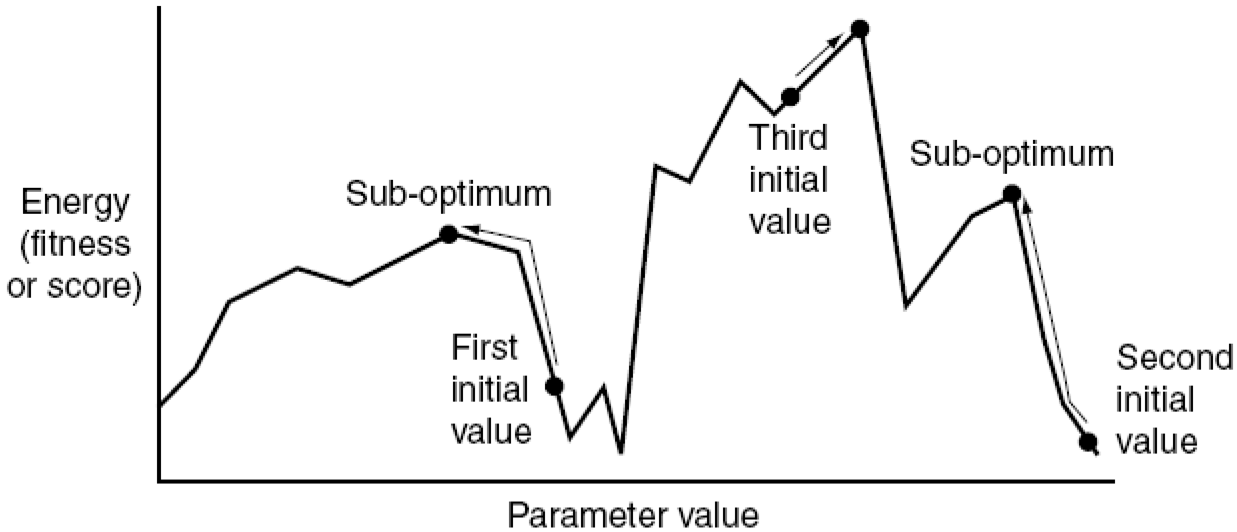
from "Artificial Intelligence for Games" by I. Millington & J. y ge

Adaptation of the Parameters

- Extensions to Hill Climbing to overcome local minima problem

  - Multiple trials: try random initial locations

Disadvantages, in particular in the context of games?



from “Artificial Intelligence for Games” by I. Millington & J. y ge

Adaptation of the Parameters

- Extensions to Hill Climbing to overcome local minima problem

  - Simulated Annealing: the selection of Δ *is done at random*

Start with the best guess for the parameter vector Ψ
Until no further improvement
    pick a random direction Δ
    With probability P, set $\Psi = \Psi + \Delta$, *where P is a function of the improvement* $(f(\Psi+\Delta)-f(\Psi))$ *and temperature T*

Adaptation of the Parameters

- Extensions to Hill Climbing to overcome local minima problem

  - Simulated Annealing: the selection of Δ *is done at random*

Start with the best guess for the parameter vector Ψ

Until no further improvement

pick a random direction Δ

With probability P, set $\Psi = \Psi + \Delta$, where P is a function of the improvement ($f(\Psi+\Delta)-f(\Psi)$) and temperature T

P is increasing as $f(\Psi+\Delta)-f(\Psi)$ increases
P is decreasing as T decreases

Temperature T is being decreased over time

Disadvantages?

As T gets closer to 0 (T "cools off"), the algorithm becomes greedy descent

Action Prediction

- Predict future action of the player based on past actions and anything else relevant

Any ideas how to do it?

Action Prediction

- N-gram predictor

  - **Very** popular in all the combat (martial arts, boxing, swords, …) games (e.g., predicting next move)

  - Often requires reduction in the power of AI to make it beatable

  - Extends to other predictions such as what weapons will be used or how the attacks will occur

Action Prediction

- N-gram predictor
  - *Maintain the probabilities of future actions based on N-1 preceding observations*
  - *For prediction, always return the most likely action based on last N-1 observations*

Action Prediction

- N-gram predictor

  - *Maintain the probabilities of future actions based on N-1 preceding observations*

  - *For prediction, always return the most likely action based on last N-1 observations*

Example of applying 3-gram predictor:

training data (observed sequence of moves): LRRLRLLLRRLRLRR

learnt prediction table:

predicted actions

	..R	..L
LL	$\frac{1}{2}$	$\frac{1}{2}$
LR	$\frac{3}{5}$	$\frac{2}{5}$
RL	$\frac{3}{4}$	$\frac{1}{4}$
RR	$\frac{0}{2}$	$\frac{2}{2}$

N-1 preceding observations

There were 5 LRs
3 resulted in R
2 resulted in L

Action Prediction

- N-gram predictor

  - *Maintain the probabilities of future actions based on N-1 preceding observations*

  - *For prediction, always return the most likely action based on last N-1 observations*

Example of applying 3-gram predictor:

training data (observed sequence of moves): LRRLRLLLRRLRLRR

learnt prediction table:

	..R	..L
LL	$\frac{1}{2}$	$\frac{1}{2}$
LR	$\frac{3}{5}$	$\frac{2}{5}$
RL	$\frac{3}{4}$	$\frac{1}{4}$
RR	$\frac{0}{2}$	$\frac{2}{2}$

Mathematically, what are you learning?

Action Prediction

- N-gram predictor

  - *Maintain the probabilities of future actions based on N-1 preceding observations*

  - *For prediction, always return the most likely action based on last N-1 observations*

Example of applying 3-gram predictor:

training data (observed sequence of moves): LRRLRLLLRRLRLRR

learnt prediction table:

	..R	..L
LL	$\frac{1}{2}$	$\frac{1}{2}$
LR	$\frac{3}{5}$	$\frac{2}{5}$
RL	$\frac{3}{4}$	$\frac{1}{4}$
RR	$\frac{0}{2}$	$\frac{2}{2}$

Mathematically, what are you learning?

P(A=L|LL),
P(A=L|LR),
...

Action Prediction

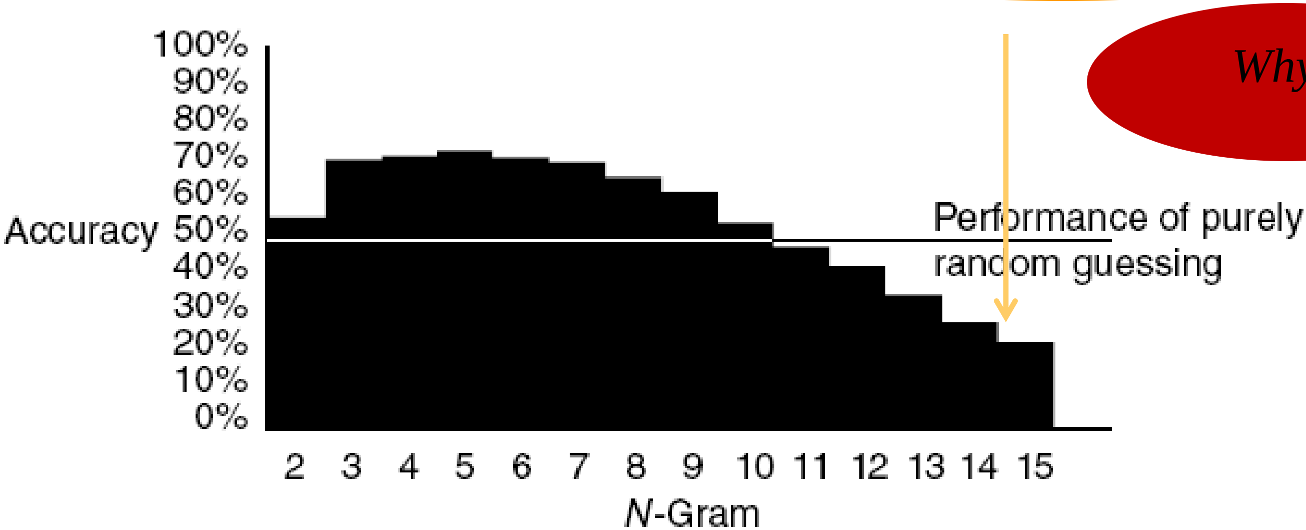
- N-gram predictor

  - *Maintain the probabilities of future actions based on N-1 preceding observations*

  - *For prediction, always return the most likely action based on last N-1 observations*

Accuracy of prediction may reduce as N increases past some point

Why?



from "Artificial Intelligence for Games" by I. Millington & J. y ge

Action Prediction

- N-gram predictor

  - *Maintain the probabilities of future actions based on N-1 preceding observations*

  - *For prediction, always return the most likely action based on last N-1 observations*

Game Performance in initial stages: what to do before the N-gram predictor is learnt?

Action Prediction

- Hierarchical N-gram predictor
  - *Learn 2-gram, 3-gram, … predictors simultaneously*
  - *For prediction, pick N-gram predictor with largest N and sufficient number of training samples for the given input*

Game Performance in initial stages: what to do before the N-gram predictor is learnt?

Action Prediction

- Hierarchical N-gram predictor

  - *Learn 1-gram, 2-gram, 3-gram, … predictors simultaneously*

  - *For prediction, pick N-gram predictor with largest N and sufficient number of training samples for the given input*

  - *If none have sufficient samples, then output random prediction*

Action Prediction

- Hierarchical N-gram predictor
  - *Learn 1-gram, 2-gram, 3-gram, … predictors simultaneously*
  - *For prediction, pick N-gram predictor with largest N and sufficient number of training samples for the given input*
  - *If none have sufficient samples, then output random prediction*

Example:

training data (observed sequence of moves): LRRLRLLLRRLRLRR

1-gram:

L	R	# of samples
7/15	8/15	15

2-gram:

Obs.	L	R	# of samples
L	2/7	5/7	7
R	4/7	3/7	7

3-gram:

Obs.	L	R	# of samples
LL	1/2	1/2	2
LR	2/5	3/5	5
RL	1/4	3/4	4
RR	1	0	2

Action Prediction

- Hierarchical N-gram predictor
  - *Learn 1-gram, 2-gram, 3-gram, … predictors simultaneously*
  - *For prediction, pick N-gram predi… …est N and sufficient number of training sa…*
  - *If none have sufficient sampl…*

Suppose we want to have at least 4 samples for prediction, what is the predicted next action for input=RRR?

Example:

training data (observed sequence of moves): LRRLRLLLRRLRLRR

1-gram:

L	R	# of samples
7/15	8/15	15

2-gram:

Obs.	L	R	# of samples
L	2/7	5/7	7
R	4/7	3/7	7

3-gram:

Obs.	L	R	# of samples
LL	1/2	1/2	2
LR	2/5	3/5	5
RL	1/4	3/4	4
RR	1	0	2

Action Prediction

- Hierarchical N-gram predictor
  - *Learn 1-gram, 2-gram, 3-gram, … predictors simultaneously*
  - *For prediction, pick N-gram predi … N and sufficient number of training sa…*
  - *If none have sufficient sampl…*

Suppose we want to have at least 9 samples for prediction, what is the predicted next action for input=RRR?

Example:

training data (observed sequence of moves): LRRLRLLLRRLRLRR

1-gram:

L	R	# of samples
7/15	8/15	15

2-gram:

Obs.	L	R	# of samples
L	2/7	5/7	7
R	4/7	3/7	7

3-gram:

Obs.	L	R	# of samples
LL	1/2	1/2	2
LR	2/5	3/5	5
RL	1/4	3/4	4
RR	1	0	2

Action Prediction

- Hierarchical N-gram predictor
  - *Learn 1-gram, 2-gram, 3-gram, … predictors simultaneously*
  - *For prediction, pick N-gram predictor with largest N and sufficient number of training samples f*
  - *If none have sufficient samples, th*

N-gram based prediction is still hard to scale with large # of possibly relevant observations

Example:

training data (observed sequence of moves): LRRLRLLLRRLRLRR

1-gram:

L	R	# of samples
7/15	8/15	15

2-gram:

Obs.	L	R	# of samples
L	2/7	5/7	7
R	4/7	3/7	7

3-gram:

Obs.	L	R	# of samples
LL	1/2	1/2	2
LR	2/5	3/5	5
RL	1/4	3/4	4
RR	1	0	2

Action Prediction

- Naïve Bayes Classifiers
  - Scale much better to large # of input variables
  - Very popular (and powerful) for machine learning problems

Action Prediction

- Naïve Bayes Classifiers
  - predicts action $A^{predict} = argmax_a \, P(A{=}a) \prod_i P(X_i = u_i \mid A{=}a)$

Action Prediction

- Naïve Bayes Classifiers
  - predicts action $A^{predict} = argmax_a\, P(A=a) \prod_i P(X_i = u_i \mid A=a)$

Derivations of the above formula:

$A^{predict} = argmax_a\, P(A=a \mid X_1 = u_1 \;\; \ldots \;\; X_k = u_k)$

$A^{predict} = argmax_a\, P(A=a \quad X_1 = u_1 \;\; \ldots \;\; X_k = u_k) \,/\, P(X_1 = u_1 \;\; \ldots \;\; X_k = u_k)$

$A^{predict} = argmax_a\, P(X_1 = u_1 \;\; \ldots \;\; X_k = u_k \mid A=a) P(A=a) \,/\, P(X_1 = u_1 \;\; \ldots \;\; X_k = u_k)$

$A^{predict} = argmax_a\, P(X_1 = u_1 \;\; \ldots \;\; X_k = u_k \mid A=a) P(A=a)$ *Why?*

Assuming conditional independence of input variables given predicted output:

$A^{predict} = argmax_a\, P(A=a) \prod_i P(X_i = u_i \mid A=a)$

Action Prediction

- Naïve Bayes Classifiers
  - predicts action $A^{predict} = argmax_a\, P(A{=}a) \prod_i P(X_i = u_i \mid A{=}a)$

Example: Predicts whether the player will slow down (break) based on the distance to obstacle corner and the speed of the car

training data (previously collected):

brake?	distance	speed
Y	near	slow
Y	near	fast
N	far	fast
Y	far	fast
N	near	slow
Y	far	slow
Y	near	fast

Suppose our current input is:
Distance = far, Speed = slow

Will character break?

Action Prediction

- Naïve Bayes Classifiers
  - predicts action $A^{predict} = argmax_a\, P(A=a) \prod_i P(X_i = u_i \mid A=a)$

Example: Predicts whether the player will slow down (break) based on the distance to obstacle corner and the speed of the car

training data (previously collected):

brake?	distance	speed
Y	near	slow
Y	near	fast
N	far	fast
Y	far	fast
N	near	slow
Y	far	slow
Y	near	fast

Suppose our current input is:
Distance = far, Speed = slow

Will character break?

$P(break) = 5/7;$
$P(Dist=far|break)=2/5;\ P(Speed=slow|break) = 2/5;$
$P(A=a) \prod_i P(X_i = u_i \mid A=a) = 5/7*2/5*2/5=4/35;$

$P(not\ break) = 2/7;$
$P(Dist=far|not\ break)=1/2;\ P(Speed=slow|not\ break) = 1/2;$
$P(A=a) \prod_i P(X_i = u_i \mid A=a) = 2/7*1/2*1/2=1/14;$

Action Prediction

- Naïve Bayes Classifiers
  - predicts action $A^{predict} = argmax_a\, P(A{=}a) \prod_i P(X_i = u_i \mid A{=}a)$

Example: Predicts whether the player will slow down (break) based on the distance to obstacle corner a

The character is more likely to break (4/35 > 1/14)

training data (previously collected):

brake?	distance	speed
Y	near	slow
Y	near	fast
N	far	fast
Y	far	fast
N	near	slow
Y	far	slow
Y	near	fast

Suppose our current input is:

Distance = far, Speed = slow

Will character break?

P(break) = 5/7;

P(Dist=far|break)=2/5; P(Speed=slow|break) = 2/5;

$P(A{=}a) \prod_i P(X_i = u_i \mid A{=}a) = 5/7*2/5*2/5=4/35;$

P(not break) = 2/7;

P(Dist=far|not break)=1/2; P(Speed=slow|not break) = 1/2;

$P(A{=}a) \prod_i P(X_i = u_i \mid A{=}a) = 2/7*1/2*1/2=1/14;$