

Procedural Content Generation: Player Models

2018-04-03

OOB

- HW 7: SMB level generation (papers added to t-square)
- Trajectory update
 - Intro: Game AI vs Academic AI; Graphs + Search
 - Taking action in a game:
 - Basic Physical Acts: Paths, Movement, Steering
 - Decisions: FSMs, D&B Trees, RBS, Plan, Meta {BB, Fuzzy}
 - **Creating & Adapting content (sans NPC decisions):**
 - **PCG, Player Modeling**
 - Adapting NPC decisions (& content) → Learning
 - Support Technologies; Design GAI; AI Based Games; Special topics (e.g. narrative); Revisits

PROCEDURAL CONTENT GENERATION: RECAP

Procedural Content Generation

- Costs exponentially increasing: levels, maps, tracks, missions, characters, weapons...
- CONTENT IS KING
- Use of computation to produce elements of gameplay (instead of manual effort)
 - Design aspects of the game
 - Save development cost
 - Save storage or main memory (“infinite games”)
 - Adapt aspects of the game (player models)

PCG as Local Search

1. What “search” is happening? Do we seek a path to goal?
2. What is the state space? How many states do we save?
3. How memory efficient is this search?
4. Hill climbing: (PCG as parameter search part 1)
 1. L___ search
 2. What is the “landscape”?
 3. Need a function that maps p___ to f_____
5. GAs: (PCG as parameter search part 2)
 1. Good in _____ domains, where _D.K._ is scarce or hard to encode
 2. Can also be used for ___ search
 3. Also needs a f_____ function (maps c___ to f_____)
6. Other local search techniques
 1. Gradient Descent
 2. Simulated annealing
 3. Local beam
 4. Tabu
 5. Ant Colony Optimization

PCG and GAs

1. Create a random set of n chromosomes (**population**)
 2. Assign a fitness score to each chromosome (**fitness function**)
 3. Remove the $m\%$ ($m < 100$) worst chromosomes
 4. Cycle through remaining pairs of chromosomes and *cross-over* (with some probability)
 5. Randomly mutate (during?) cross-over (with some probability)
 6. Reduce new population to size n
 7. Repeat steps 2-6 until [stepwise improvement diminishes || one individual is fit enough || # generations reached]
- **Tuning parameters**
 - Population size
 - Number of generations
 - Fitness function
 - Representation
 - Mutation rate
 - Crossover operations
 - Selection procedure
 - Number of solutions to keep

PLAYER MODELING

Player Modeling

A set of approaches that represent/quantify the player's experience or preferences with the goal of improving future experiences.

Improving the experience either:

1. Within the game, altering game elements or deciding between elements (picking which Mario Kart item)
2. Outside the game, giving feedback to designers/developers (analytics)

Model-based versus Model free

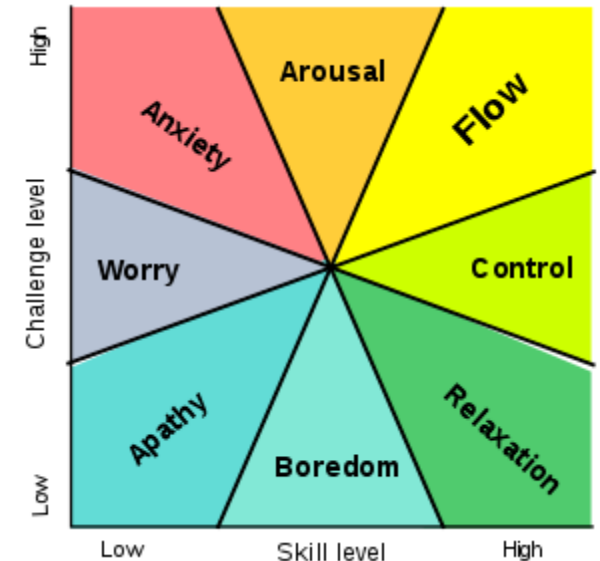
We can separate player modeling approaches into two more general groups:

- **Model-based:** Roughly corresponds to player modeling within games. It presupposes we have a set of categories or value range to assign to some part of a player's experience
- **Model free:** Roughly corresponds to player modeling outside of games. Looks for patterns of behavior without pre-existing hypothesis, drawing on statistical/machine learning techniques

Model-based: 4 Major Models

- **Note:** This is not to say that there exists only four models, but that vast majority can be understood as derivatives of these four

1. Csikszentmihalyi's Flow (Position in space)
2. Bartle's player types (Categories of play style)
3. Elo's performance rating (Continuous value)
4. Drama Management (Difference between target value and .)



Model free Approaches

- Start with some data about a playerbase and game world
- **Categorize** learn what types exist
- **Regression analysis** how does x relate to y ?
 - X : Input about the playerbase/game world
 - Y : Something we want to predict/understand
- **Classify** which y is x a member of?

MODEL-BASED PCG

Player Modeling

- What is optimal in PCG?
- Need a player model
 - Tells you something about the player
 - Makes predictions about player
- Challenges in player modeling:
 - What do you model?
 - Where do you get the model?
 - How do you use the model?

Player Model

- What do we want to model?

What do we want to model?

- Demographics
- Game play traces
- Stats
- Features of world when actions are performed
 - Eg: more likely to jump when X is true
- Sensors
- Preferences
 - Ask before, during, after → ratings
- Personality

Example Statistics

CAT 1: (FPS – L4D)

- Ave player health
- Shot accuracy
- # times damaged
- # times killed

CAT3: (rpg)

- Unique area visits
- Time spent exploring (straying)
- Win/loss ratio
- Time to complete level
- Items found
- Item usage

CAT2: (platformer -- mario)

- Power up blocks collected
- # deaths
- # deaths from gaps
- # times ducked
- Time between deaths
- Con blocks pressed
- # gaps jumped
- Avg width of gaps jumped

Player Model

- What do we want to model?
- Where do we get the model?

Where do we get the model?

- Observe player

Where do we get the model?

- Observe player
- Questionnaire

Where do we get the model?

- Observe player
- Questionnaire
- Social media?

Player Model

- What do we want to model?
- Where do we get the model?
- How do we use the model?

How do we use the model?

- Must be input to algorithm

How do we use the model?

- Must be input to algorithm
- Must actually make a difference

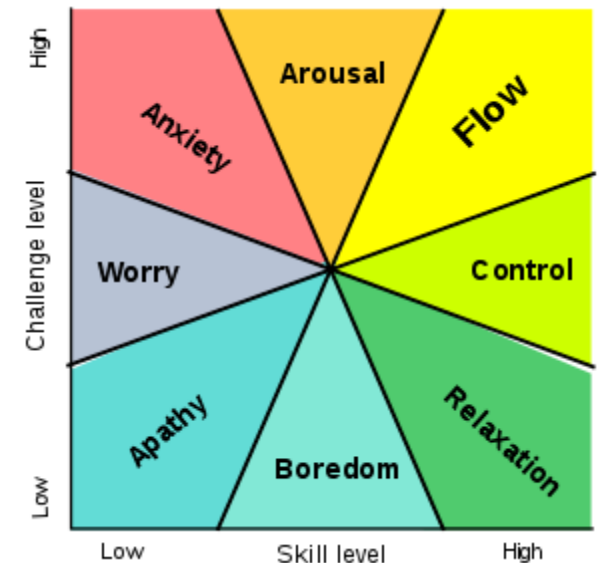
How do we use the model?

- Must be input to algorithm
- Must actually make a difference
- Machine Learning
 - Learn to make decisions based on observed data
 - Optimization seeks best mapping
 - Genetic algorithms
 - Simulated annealing

Model-based: 4 Major Models

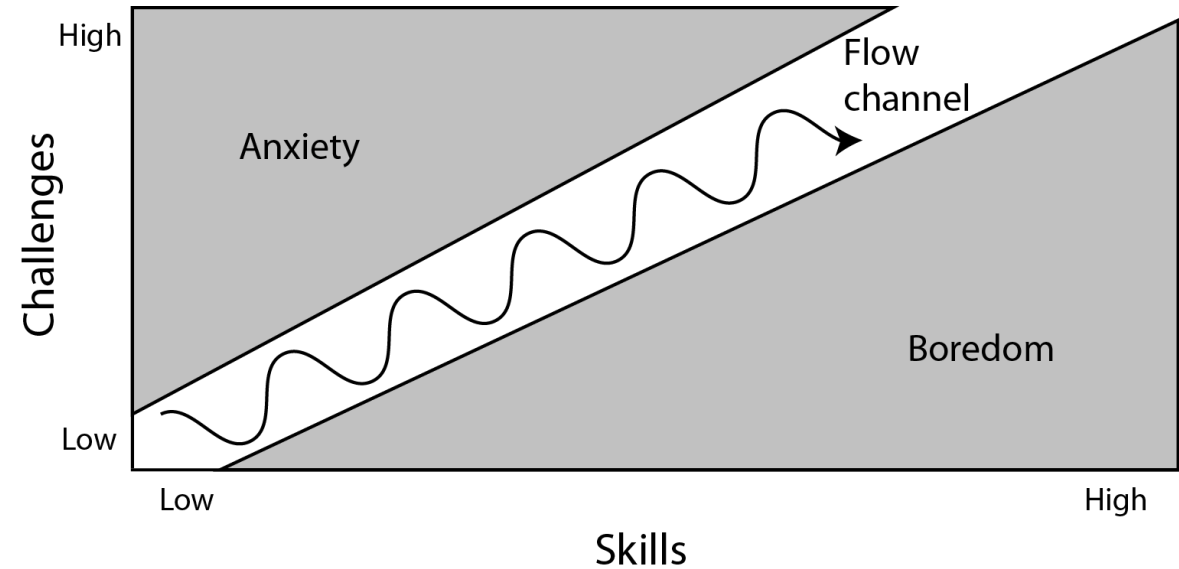
- **Note:** This is not to say that there exists only four models, but that vast majority can be understood as derivatives of these four

1. Csikszentmihalyi's Flow (Position in space)
2. Bartle's player types (Categories of play style)
3. Elo's performance rating (Continuous value)
4. Drama Management (Difference between target value and .)



FLOW

Mario Kart: Flow



Input: Skills (player placement in race), and Challenges (power-ups of enemies)

Output: A position in space.

Decision: What do we do about it?

Flow

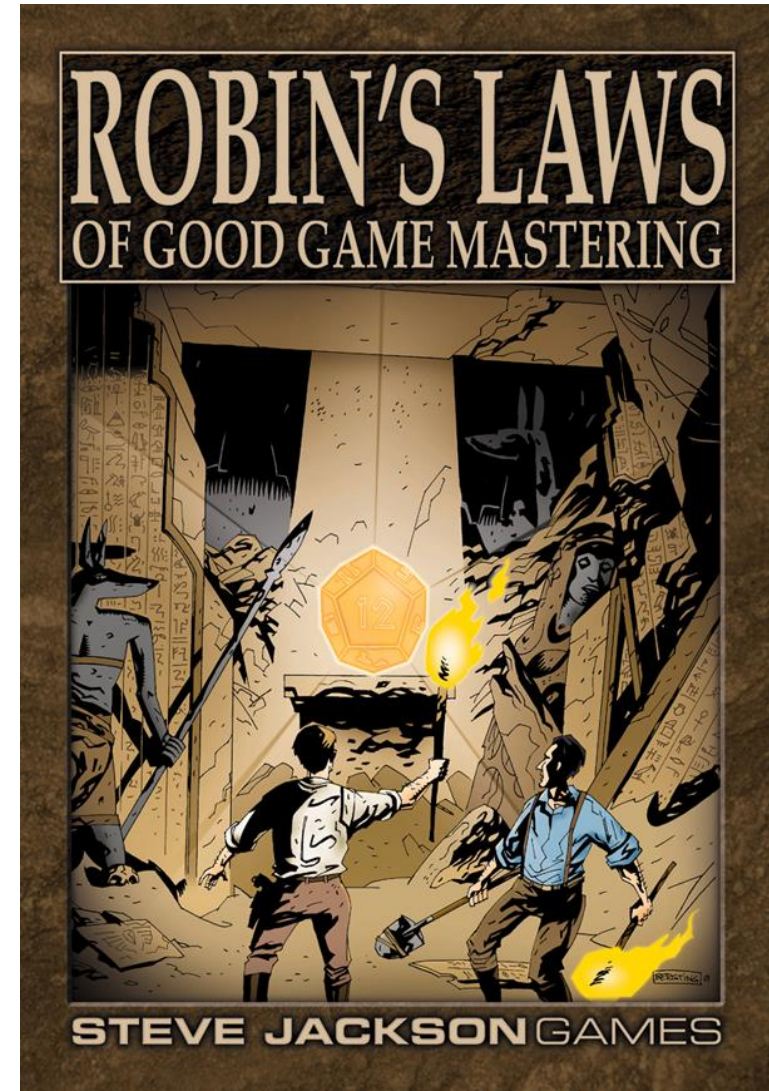
- Advantages
 - Intuitive
 - Matches player experience
- Disadvantages
 - Need to design reliable measures for player challenge and skill, not possible in all games
 - Difficult for non-competitive multiplayer games

MODELS: ROBIN, BARTLE, YEE

Robin's Laws of Good (tabletop RPG) Gamemastering

Five player archetypes:

- Fighter
 - prefers combat
- Power Gamer
 - gaining special abilities and riches
- Method Actor
 - taking dramatic actions
- Tactician
 - prefers thinking creatively & solving problems
- Storyteller
 - prefers complex plots



Player Types within a game Example: Dragon Age Prototype

Five types, what
motivates?

- Fighters (combat)
- Power Gamers (gaining special items)
- Tacticians (creative thinking)
- Storytellers (complex plot)
- Actors (dramatic actions)



How it worked

- Each player has a vector
 - (Fighter, MethodActor, Storyteller, Tactician, PowerGamer)
- Every action is tagged with one or more of these roles.
- Every time a player takes an action, update the value of the vector index of the associated role(s).
- System chooses reward to incentivize the maximal role(s)

Example

- Player A has vector (F: 0, MA: 0,S: 0,T: 0,PG: 0)
- Player A asks for a reward for a quest, an action which is strongly tagged with the PowerGamer role (value of 100)
- Player A now has a vector (F: 0, MA: 0,S: 0,T: 0,PG: 100)
- Later on Player A will be offered more powerful items if their vector remains unchanged

Player Types

- Advantages
 - Can allow designers to understand player base
 - Can broaden the appeal of a game
- Disadvantages
 - Different games/genres need their own player types designed, a tricky task
 - E.g. “Whales” in free-to-play mobile games
 - In-game usage of this model requires a lot of extra knowledge authoring

Robin's Laws of Good Gamemastering

- Suppose actions, decisions, and/or choice points are tagged with types and strengths
 - e.g. Action X has *method-actor* = 8
- Player model is vector of accumulated strengths (normalized?)
 - $\langle F, PG, MA, T, S \rangle$
 - $\langle 1, 141, 81, 1, 1 \rangle$
- Incrementally build model

Robin's Laws of Good Gamemastering

- How does world/actions translate to model?

Robin's Laws of Good Gamemastering

- How does world/actions translate to model?
- How does model map to design?

Robin's Laws of Good Gamemastering

- How does world/actions translate to model?
- How does model map to design?

ACTIONS ---> MODEL ---> ACTIONS

FEATURES ---> MODEL ---> FEATURES

Robin's Laws of Good Gamemastering

- How does world/actions translate to model?
- How does model map to design?

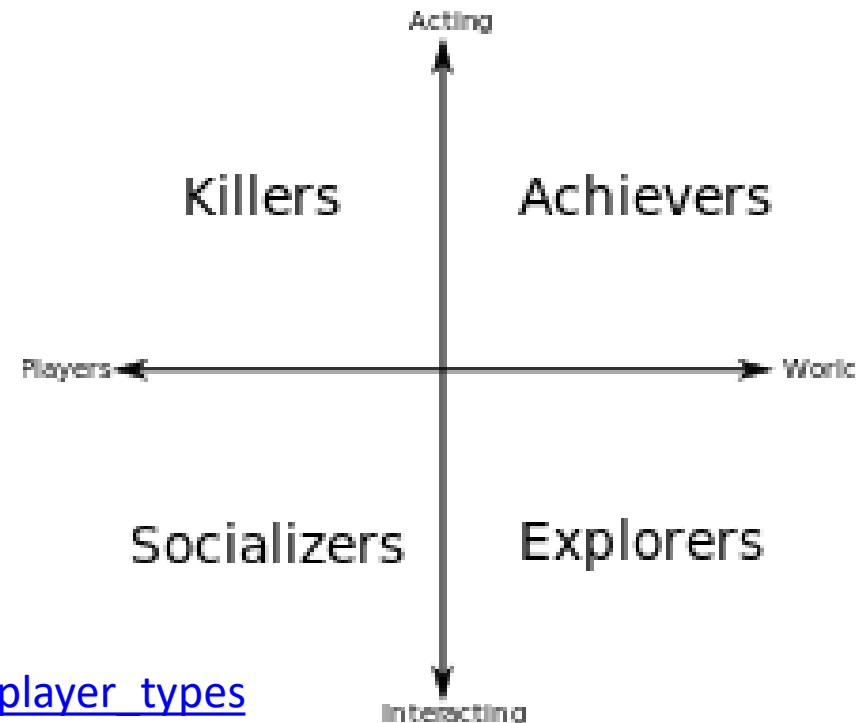
ACTIONS ---> MODEL ---> ACTIONS

FEATURES ---> MODEL ---> FEATURES

- Player simulation
- See <http://www.sjgames.com/robinslaws/>

Bartle: MUD Player Modeling

- MUD = Multi-user Dungeon
- 4 things people typically enjoy doing in MUDs:
 - Achievement (preset in-game goals eg kill, aquire)
 - Exploration
 - Socializing
 - Killers (Imposing on other players)



Bartle's Player Types

- Most commonly used today in the design of MMORPG's, to motivate these different "types" of players
- Example: World of Warcraft
 - Experience for killing mobs (killer)
 - completing quests (achievers)
 - exploring (explorers)
 - battlegrounds (socializers)



MUD Player Modeling

- Everyone has a primary type
 - Bit of each, tendency towards one
- Take the Bartle Test!
- Suggests a feature vector
 - <killer, achiever, explorer, socializer>*
 - < 0.0, 0.7, 0.1, 0.2 >*
- See (no really, follow the 1st link)
 - <http://www.mud.co.uk/richard/hclds.htm>
 - http://en.wikipedia.org/wiki/Bartle_Test

Yee Player Model

- Used iterative process to validate, expand and refine a player motivation model **empirically**
- Based on open-ended questions about motivation
 - *“I feel powerful in the game.”*
 - *“I like to be immersed in a fantasy world.”*
- See
 - <http://www.nickyee.com/daedalus/archives/001298.php>

Yee Player Model: Factors

- Relationship
- Manipulation
- Immersion
- Escapism
- Achievement

“Performed factor analysis on data to separate the statements into clusters where items within each cluster were as highly correlated as possible while clusters themselves were as uncorrelated as possible” – Yee

YEE Player Model

Achievement	Social	Immersion
Advancement Progress, Power, Accumulation, Status	Socializing Casual Chat, Helping Others, Making Friends	Discovery Exploration, Lore, Finding Hidden Things
Mechanics Numbers, Optimization, Templating, Analysis	Relationship Personal, Self-Disclosure, Find and Give Support	Role-Playing Story Line, Character History, Roles, Fantasy
Competition Challenging Others, Provocation, Domination	Teamwork Collaboration, Groups, Group Achievements	Customization Appearances, Accessories, Style, Color Schemes
		Escapism Relax, Escape from RL, Avoid RL Problems

<http://www.nickyee.com/daedalus/archives/001298.php?page=4>

Motivations do not “suppress”

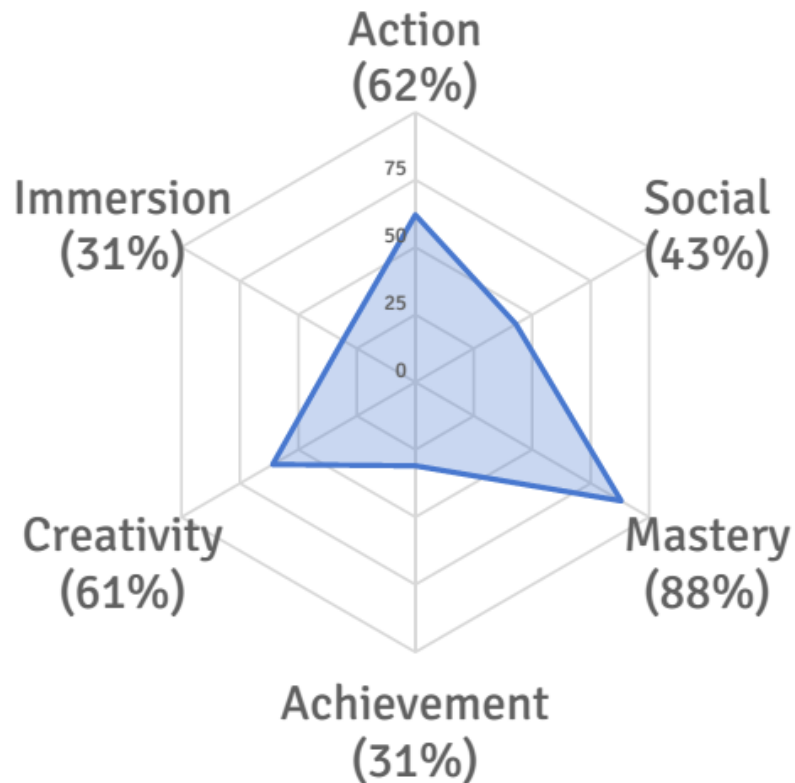
“The subcomponents generated by the factor analysis are NOT player types.

- It is NOT the case that we have come up with 10 boxes that we can put players in, but rather, we have revealed 10 subcomponents that co-exist and together reveal the motivations of a player.
- Bartle assumed that your underlying motivations "suppressed" each other. In other words, the more of an Achiever you were, the less of a Socializer, Explorer and Killer you could be, but just because you like ice-cream doesn't mean you will hate pasta.
- The assumption of polarized motivations is also not supported by the correlations of the current data set.”

Subcomponent	Inventory Item	Factor Loading
Advancement $\alpha = .79$	Leveling up your character as fast as possible.	.68
	Acquiring rare items that most players will never have.	.77
	Becoming powerful.	.81
	Accumulating resources, items or money.	.69
	How important is it to you to be well-known in the game?	.53
	Being part of a serious, raid/loot-oriented guild.	.60
Mechanics $\alpha = .68$	How interested are you in the precise numbers and percentages underlying the game mechanics?	.78
	How important is it to you that your character is as optimized as possible for their profession / role?	.65
	How often do you use a character builder or a template to plan out your character's advancement at an early level?	.67
	Knowing as much about the game mechanics and rules as possible.	.69
	Competing with other players.	.64
Competition $\alpha = .75$	How often do you purposefully try to provoke or irritate other players?	.81
	Dominating/killing other players.	.72
	Doing things that annoy other players.	.82
Socializing $\alpha = .74$	Getting to know other players.	.82
	Helping other players.	.65
	Chatting with other players.	.77
Relationship $\alpha = .80$	Being part of a friendly, casual guild.	.63
	How often do you find yourself having meaningful conversations with other players?	.71
	How often do you talk to your online friends about your personal issues?	.88
	How often have your online friends offered you support when you had	.86

Where do you fit?

- Proficient, Relaxed, Competitive, Grounded, and Inquisitive



After filling out a brief survey (5-7 minutes), this profile tool will generate a customized report and a list of recommended games for you. The report will describe the traits that were measured, and how you compare with other gamers.

<https://apps.quantificfoundry.com/lab/gamerprofile/>

ELO



 OVERWATCH RANKED DIVISIONS			
ICON	NAME	SKILL RATING	SEASON 1
	BRONZE	1-1499	1-29
	SILVER	1500-1999	30-39
	GOLD	2000-2499	40-49
	PLATINUM	2500-2999	50-59
	DIAMOND	3000-3499	60-69
	MASTER	3500-3999	70-79
	GRANDMASTER	4000-5000	80-100
	TOP 500	TOP 500 IN REGION	

Elo's Performance Rating

- Algorithm:
 - For each win, add your opponent's rating plus 400
 - For each loss, add your opponent's rating minus 400
 - Divide this sum by the number of played games
- Formula:
$$\frac{\text{Total of opponents' ratings} + 400(\text{Wins} - \text{Losses})}{\text{Games}}$$

Quick Note: Elo rating naming confusion

Many games do not make use of this exact rating algorithm, but you will still hear references to “Elo rating”/Elo hell

At a high level this phrase refers to any approach to update player skills given wins/losses and prior player skills

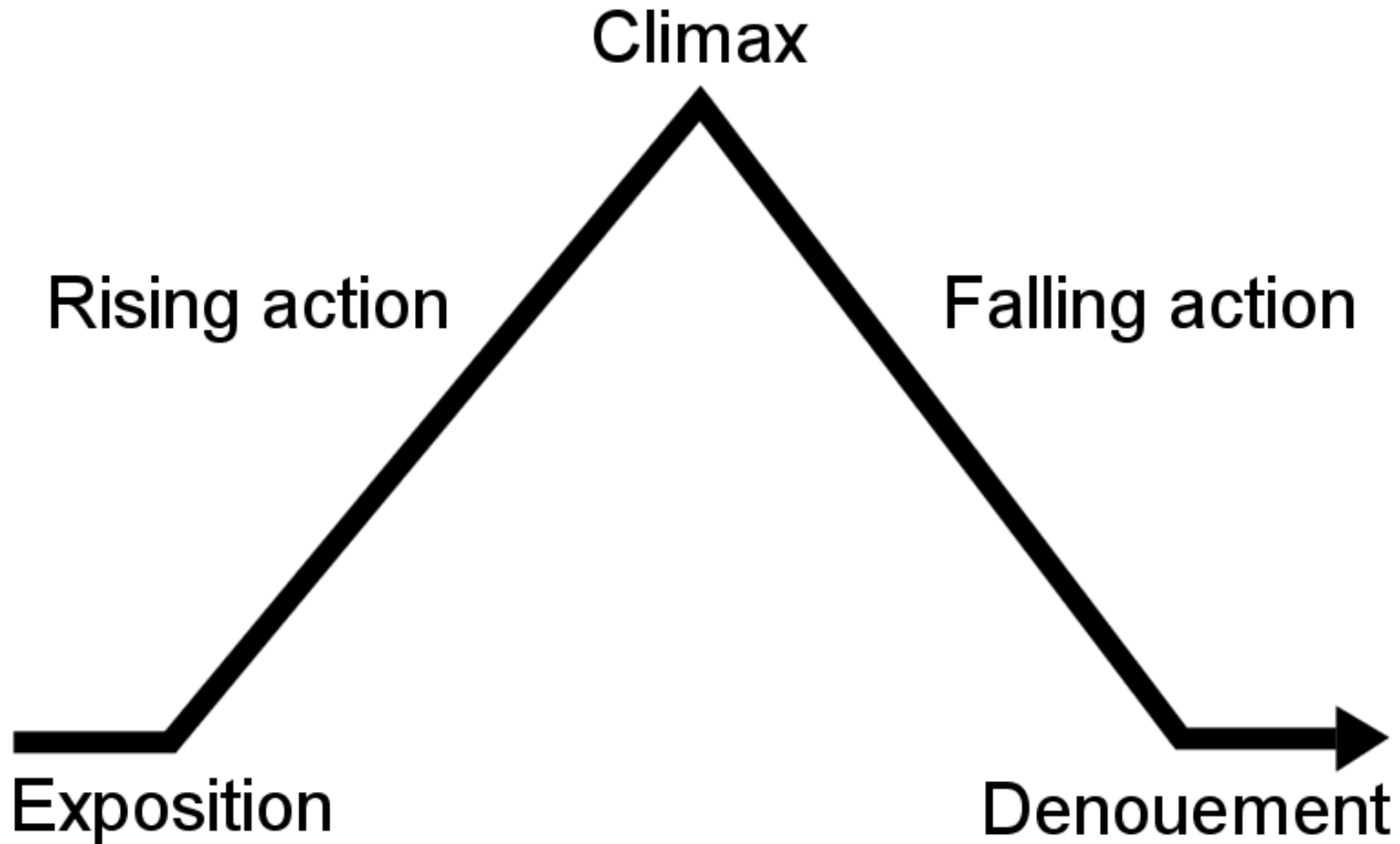
- Probability distribution

Elo Rating

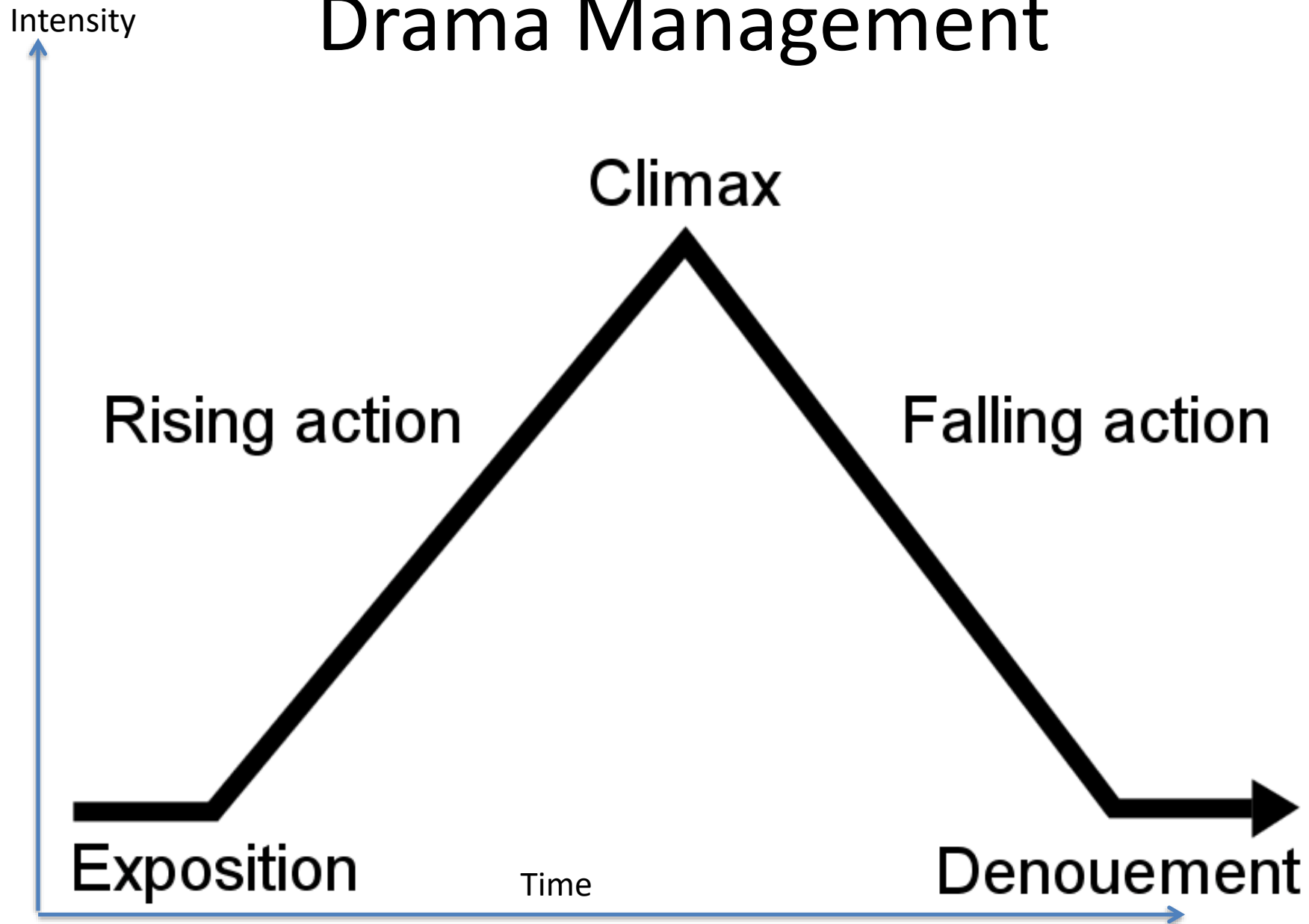
- Advantages
 - Requires very little designer authoring
 - Does a surprisingly good job of keeping competitive games fair
- Disadvantages
 - Player's do not like seeing this value drop
 - Solution: Many games have “hidden” true Elo ratings
 - Limited to competitive experiences of 2 players
 - Only makes sense when comparing players to others

DRAMA MANAGEMENT

Drama Management



Drama Management



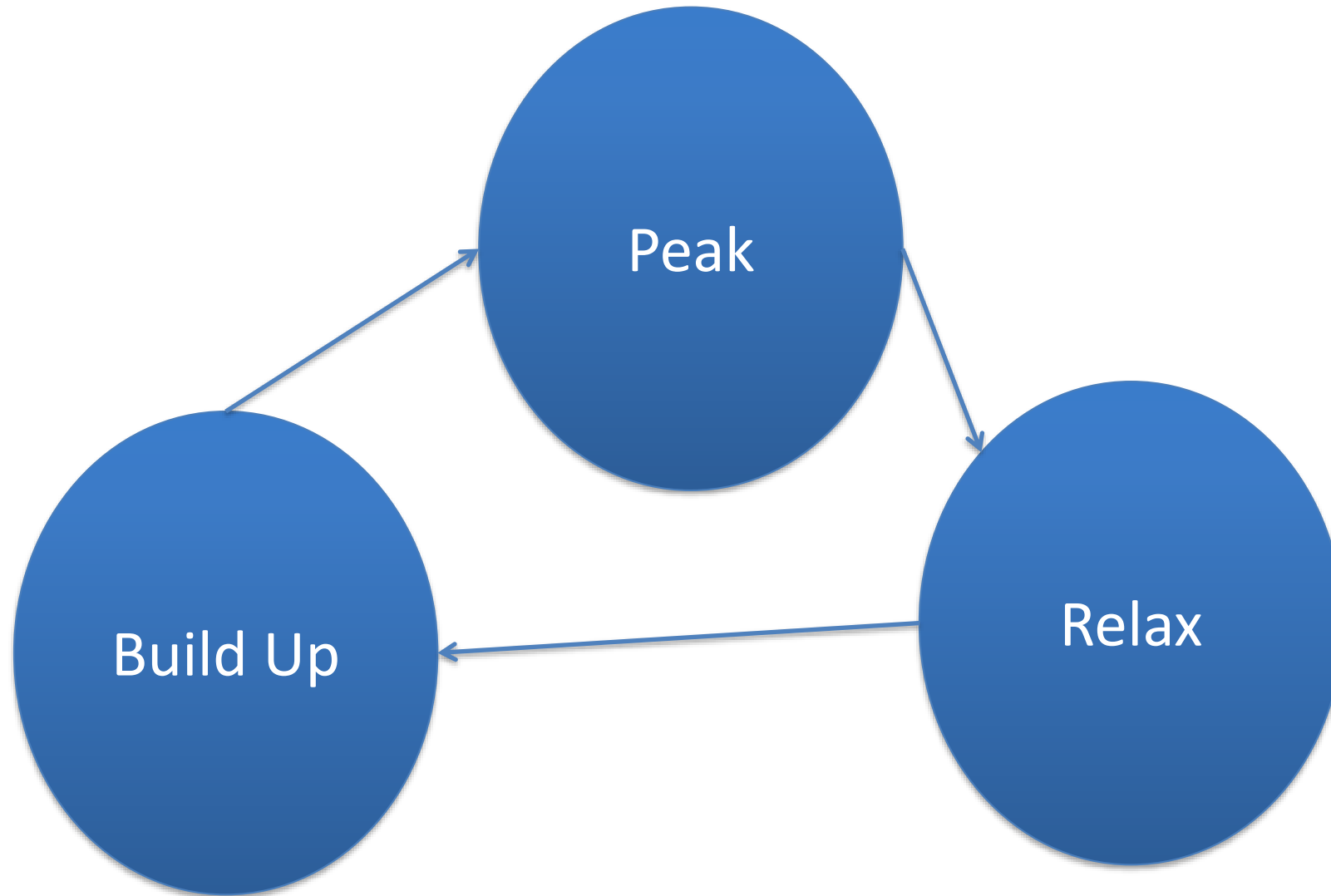
Left4Dead AI Director

- Constantly measures player “intensity”
 - Intensity increases when attacked/killing
 - Players hit max when incapacitated
 - Players slowly drop intensity when nothing is happening and reset at the start of a new stage
 - (This is a lot like challenge in flow)

Left4Dead AI Director

- Three “modes”
 - **Build up:** AI Director attempts to increase tension
 - **Peak:** Intensity reaches max, director stops spawning enemies
 - **Relax:** Director spawns helpful items/alters path to decrease tension

AI Director as FSM

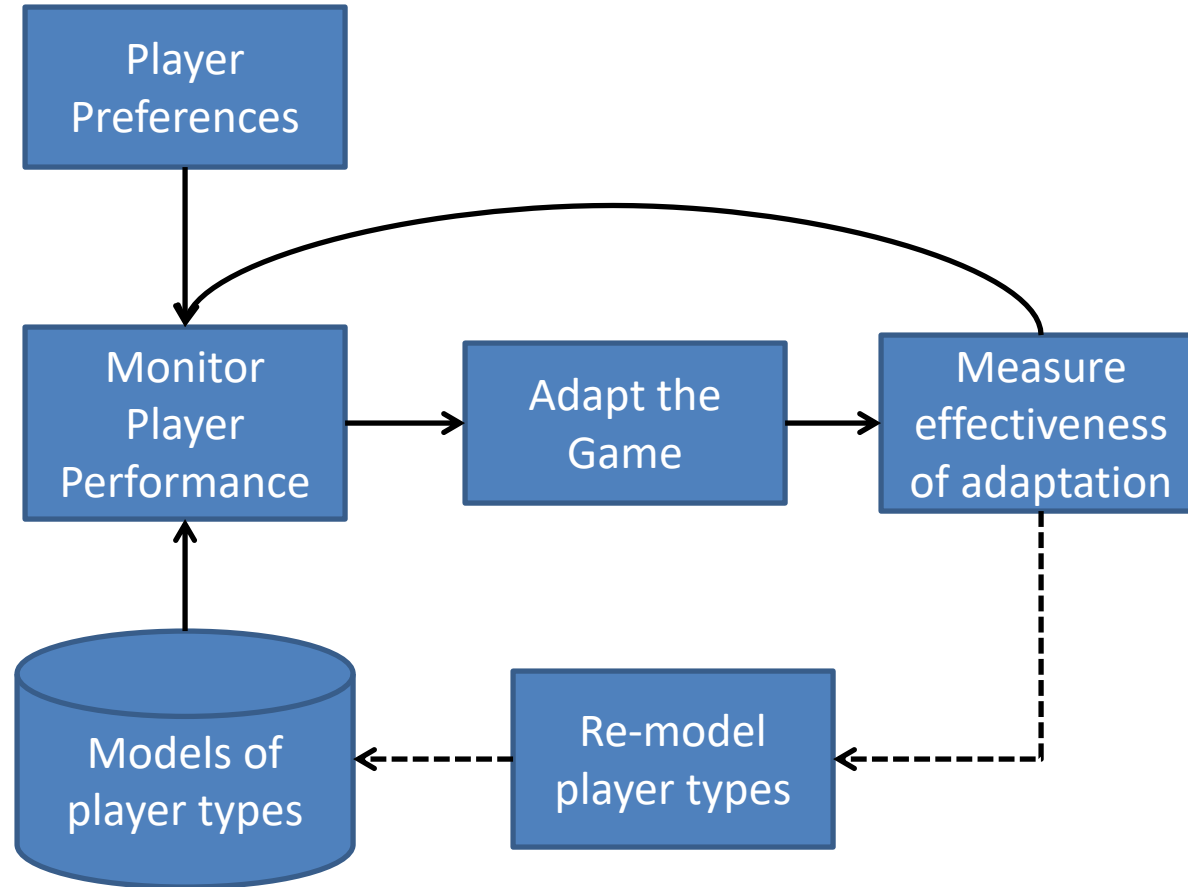


Drama Management Summary

- Advantages
 - Most effective, highly praised
 - Gives players another level of strategy
- Disadvantages
 - Fair amount of authoring (less than Player Types, more than Flow)
 - All the downsides of FSMs

PCG: STEPPING BACK

Player Model Process



Concerns

Concerns

- Player desires

Concerns

- Player desires
- Privacy

Concerns

- Player desires
- Privacy
- Testing

Concerns

- Player desires
- Privacy
- Testing
- Player experience

Summary

Authoring burden for using in a game:
player types > drama management > flow > ELO

Ultimately: Decision of when to use what depends on the type of game and desired player experience

THOUGHT EXPERIMENT: SMB LEVEL GENERATION

Mario Level Generation

- Predict users' emotional response to level:
 - Fun
 - Challenging
 - Boring
 - Frustrating

Mario Level Generation

- Controllable features

Mario Level Generation

- Controllable features
- {#gaps, gap_width, spatial_diversity}

Mario Level Generation

- Controllable features
- {#gaps, gap_width, spatial_diversity}
- Fixed: #coins, #enemies, # ?-boxes, #powerups, etc.

Building the Model

- Option #1: Pairwise Ordering
- For each pair of levels, ask:
 - Level A is more <adj> than Level B*
 - Levels A and B are equally <adj>*
 - Neither Level A nor Level B are <adj>*

Where <adj> = {fun, challenging, boring, frustrating}

Building the Model

- Option #2: Use ANN to learn to predict question-answering

Building the Model

- Option #2: Use ANN to learn to predict question-answering
- Need a function that maps features to a measure of quality:
 $f(\text{gaps, width, spatial_diversity}) \rightarrow \mathbf{R}$

Building the Model

- Other option: Learn decision tree with continuous variables
- Benefit of trained neural network: The network ***IS*** your fitness function.

Using the Model

- Search for optimal feature set for a player

Using the Model

- Search for optimal feature set for a player
- State is 3-tuple $\{G, W, SD\}$ – Could brute-force

Using the Model

- Search for optimal feature set for a player
- State is 3-tuple $\{G, W, SD\}$ – Could brute-force
- Optimization search – Genetic algorithms, hill-climbing, simulated annealing

Smith, Treanor, Whitehead, Mateas {FDG, 2009}

RHYTHM-BASED PLATFORMER LEVEL GENERATION

Rhythm based level-generation

- Patterned set of moves

|-----|-----|-----| 20s, regular, low density

move 0 5

|-----|-----|-----|-----| 20s, regular, medium density

jump 2 2.25

|-----|-----|-----|-----|-----|-----|-----| 20s, regular, high density

jump 4 4.25

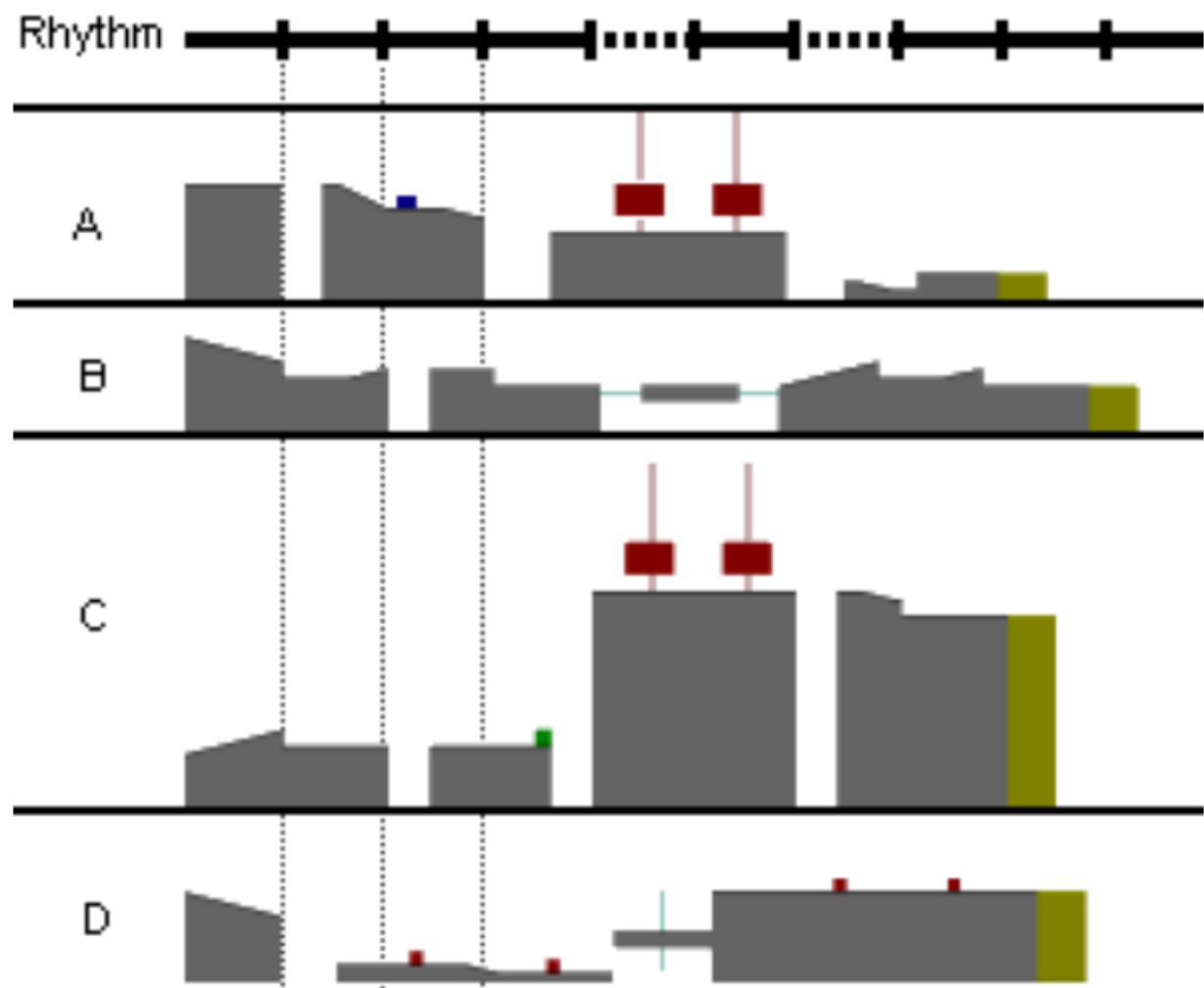
|-----|-----|-----|-----| 15s, swing, medium density

move 6 10

jump 6 6.5

|-----|-----|-----|-----| 10s, random, high density

jump 8 8.5



Geometry grammar

- Rhythm determines what roots
- Terminals determine specific content element selections

Moving → Sloped | flat_platform

Sloped → Steep | Gradual

Steep → steep_slope_up | steep_slope_down

Gradual → gradual_slope_up | gradual_slope_down

Jumping → flat_gap

| (gap | no_gap) (jump_up | Down | spring | fall)

| enemy_kill

| enemy_avoid

Down → jump_down_short | jump_down_medium | jump_down_long

Waiting-Moving → stomper

Waiting-Moving-Waiting → moving_platform_vert

| moving_platform_horiz

Grammar generation problems

- Over-generation
- Global constraints

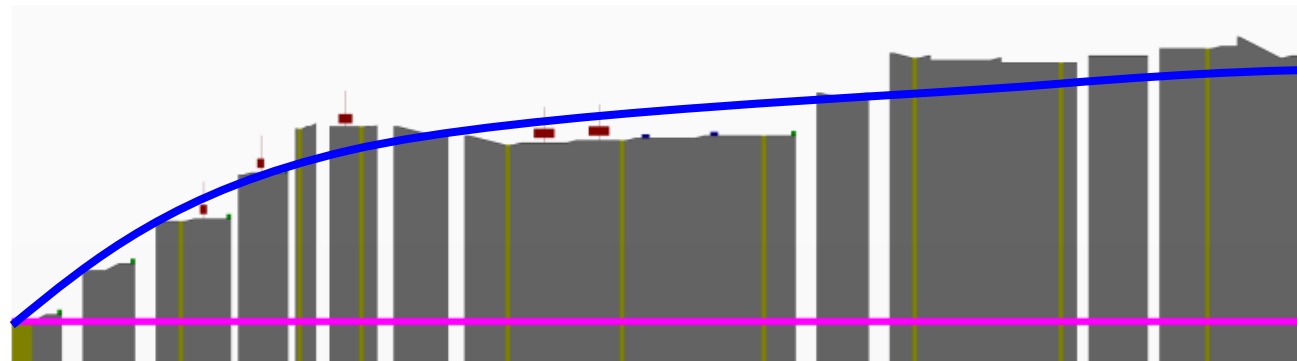
Problems with Grammars

- Problem with grammars: global context
- Can you fix grammars?
- Generate-and-test with some evaluation function
- Backtrack?
- Planning: Search for a sequence of actions that meet a goal
 - Heuristics meant to speed up search
 - Heuristics can also be used to guide algorithm toward more preferable solutions
- HTN planning: Tasks decompose to sub-tasks

Critics

- Generate-and-test: generate a bunch of levels and pick the one that scores best
- Critics are pieces of code that evaluate and select
- Line distance critic
 - Given a path the level should follow
- Component style critic
 - Consistency of components
 - Jumping caused by gaps vs. monsters

Control lines
for line critic



Credit: Matthew Guzdial

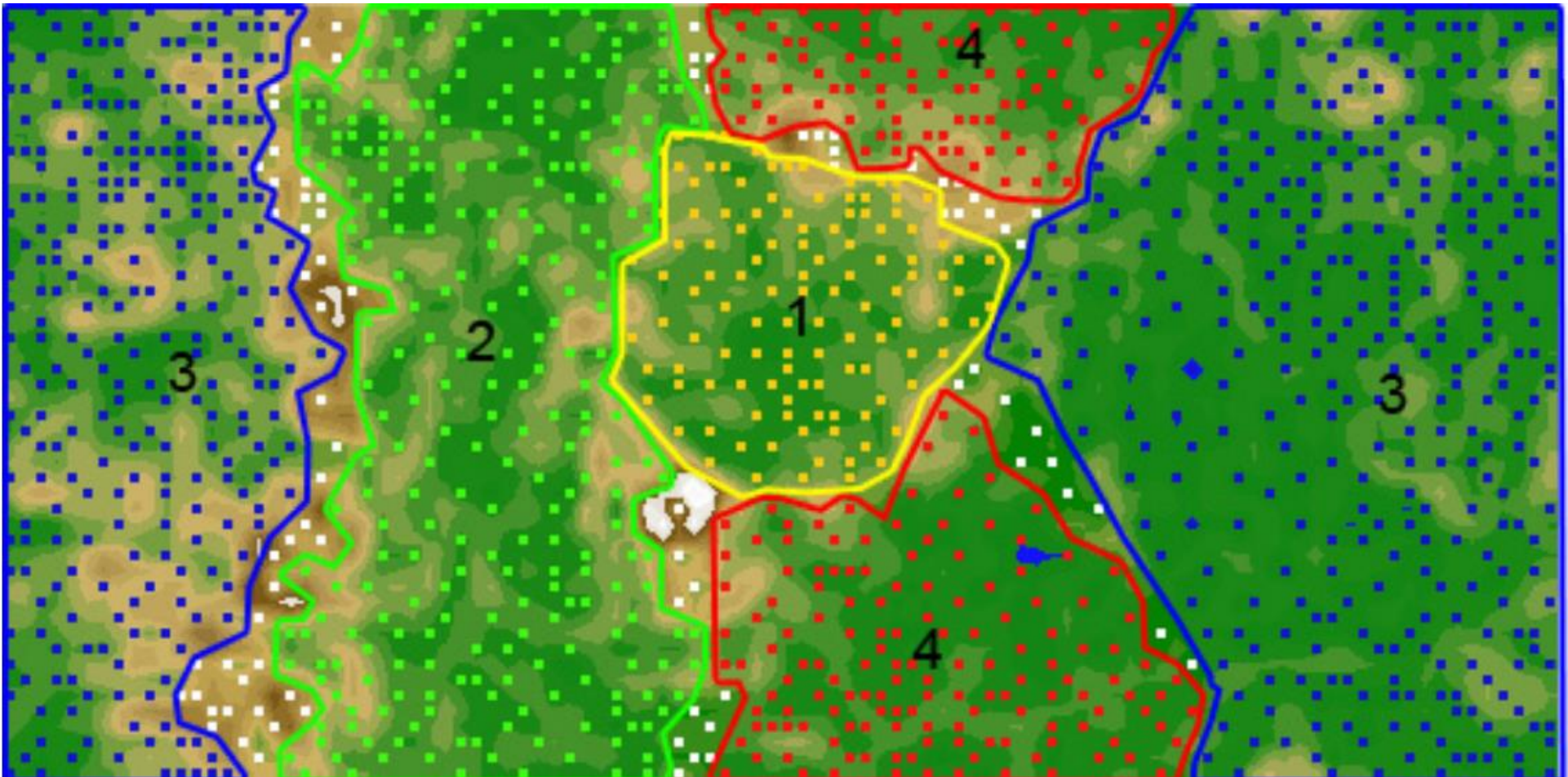
PLAYER MODELING: ANALYTICS (MODEL FREE APPROACHES)

Model free Approaches

- Start with some data about a playerbase and game world
- **Categorize** learn what types exist
- **Regression analysis** how does x relate to y ?
 - X : Input about the playerbase/game world
 - Y : Something we want to predict/understand
- **Classify** which y is x a member of?

Quick Example

<https://youtu.be/HJS-SxgXA14?t=6>



Clustering

- Uses:
 - Automatically discover categories
 - Group similar entities
 - Reduce complexity/variance
 - E.g. easier to say this player is of 4 types than 1 of 1000
- Common Approaches
 - K means/medoids
 - Hierarchical clustering
- Further info: <http://scikit-learn.org/stable/modules/clustering.html>

K means

initialize centroids randomly

oldcentroids = []

while not centroids==oldcentroids:

 oldcentroids = centroids

 calculateClusters();//cluster each element to closest centroid

 centroids = average of each cluster

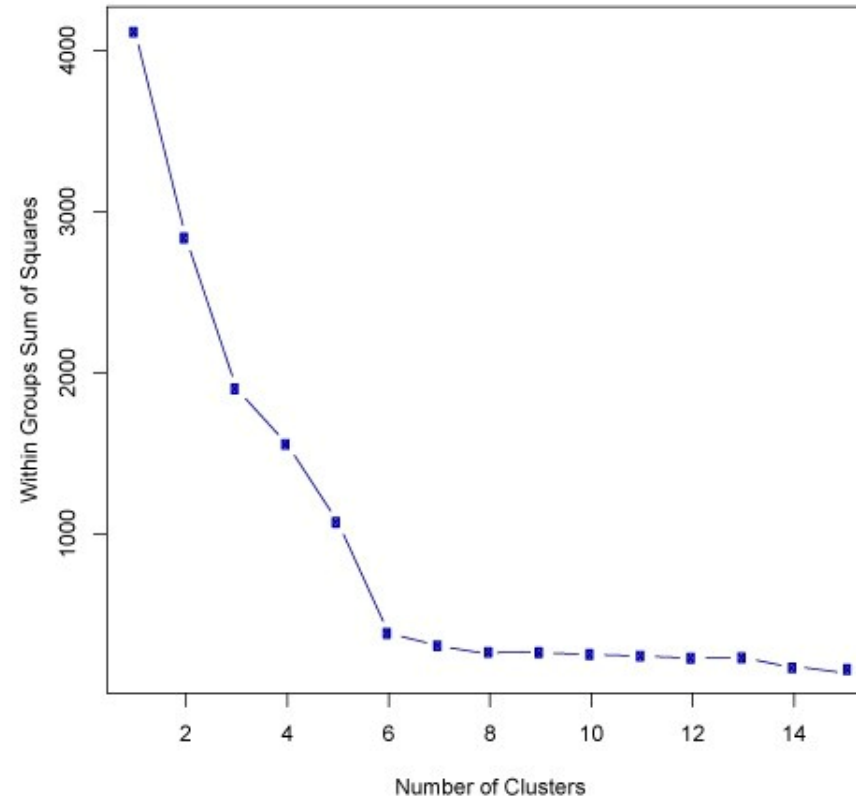
return centroids

Demonstrate on the board

(This requires a lot of erasing/updating so
doesn't really make sense as a figure)

Restrictions and solutions

- Means can give unintuitive clusters
 - K Medoids
- Have to give K
 - Elbow method



Other clustering techniques?

- There are a lot!
- But this isn't a machine learning course
- Plus...

K-Means Algorithms

Unsupervised Learning

Cluster analysis

Data Mining

Machine Learning

Why is k-means clustering so popular, given that more sophisticated alternatives exist?

1 Answer



Muktabh Mayank, Data Scientist, CoFounder @ ParallelDots, BITSian for life, love new technology trends

Answered Jan 23, 2015

The answer to any "why <a simpler Machine Learning technique> is more popular than <an advanced Machine Learning technique> is generally that the gains in performance for a complex algorithm are not worth the complexity of implementation for most people"

Open Questions

- How do we pick what variables to cluster on?
 - To normalize, or not to normalize?
- How do we pick a distance function?
- How do we use the clusters once we have them?

Question(s) 1

Think of a game and a particular element of that game (player strategy/player deaths) you'd apply clustering to.

What variables would you cluster on? Why?

What distance function would you use? Why?

How could these clusters benefit the game devs?

Clustering

- Pros:
 - Can learn “big picture” information
 - Can cut down on complexity
 - Can still give good answer with errors in training data
- Cons:
 - Individuals can get lost in the noise of groups
 - Those decisions to the open questions can hugely impact results
 - Slow to build
 - Even slower if we want to objectively decide on K
 - Slow to use

Quick Tangent: Churn

- Churn rate: The rate at which customers cut ties with a company
- In games this is how quickly a game loses players. After how long do they stop playing?
- Churn is one of the most common problems player analytics teams are tasked with

Regression

- If we can map variables onto likelihood of churn (how long the players will play) we can figure out what the designers need to fix!
- Two approaches
 - Linear regression
 - Decision trees

Only two regression techniques?

- No of course not.
- But similar story as with K means...

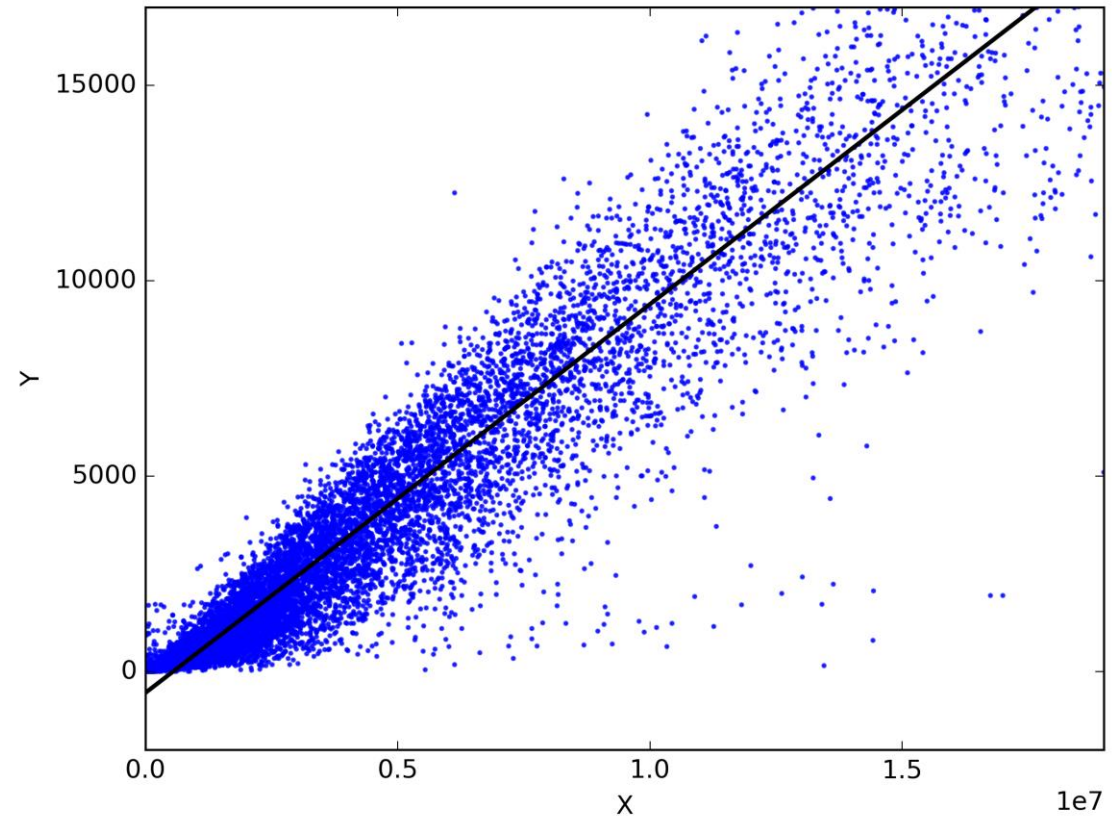
Algorithm	Level 1	Levels 1 and 2
Logistic regression	48.3	77.3
MLP/Backpropagation	47.7	70.2
J48 (C4.5) decision tree (pruned)	48.7	77.4
REPTree decision tree (pruned)	48.5	77.2
Multinomial naive bayes	43.9	50.2
Bayes network	46.7	65.1
SMO Support vector machine	45.9	70.0
Baseline	39.8	45.3

Linear Regression

Given input X and expected output Y , find m and b such that:

$$Y = m * X + b$$

It is, of course, often impossible to find exact values



Linear Regression Algorithm

Find the best fit values for m (slope) and b (y-intercept)

Lots of algorithms to do this

- Minimizing the sum of squared residuals

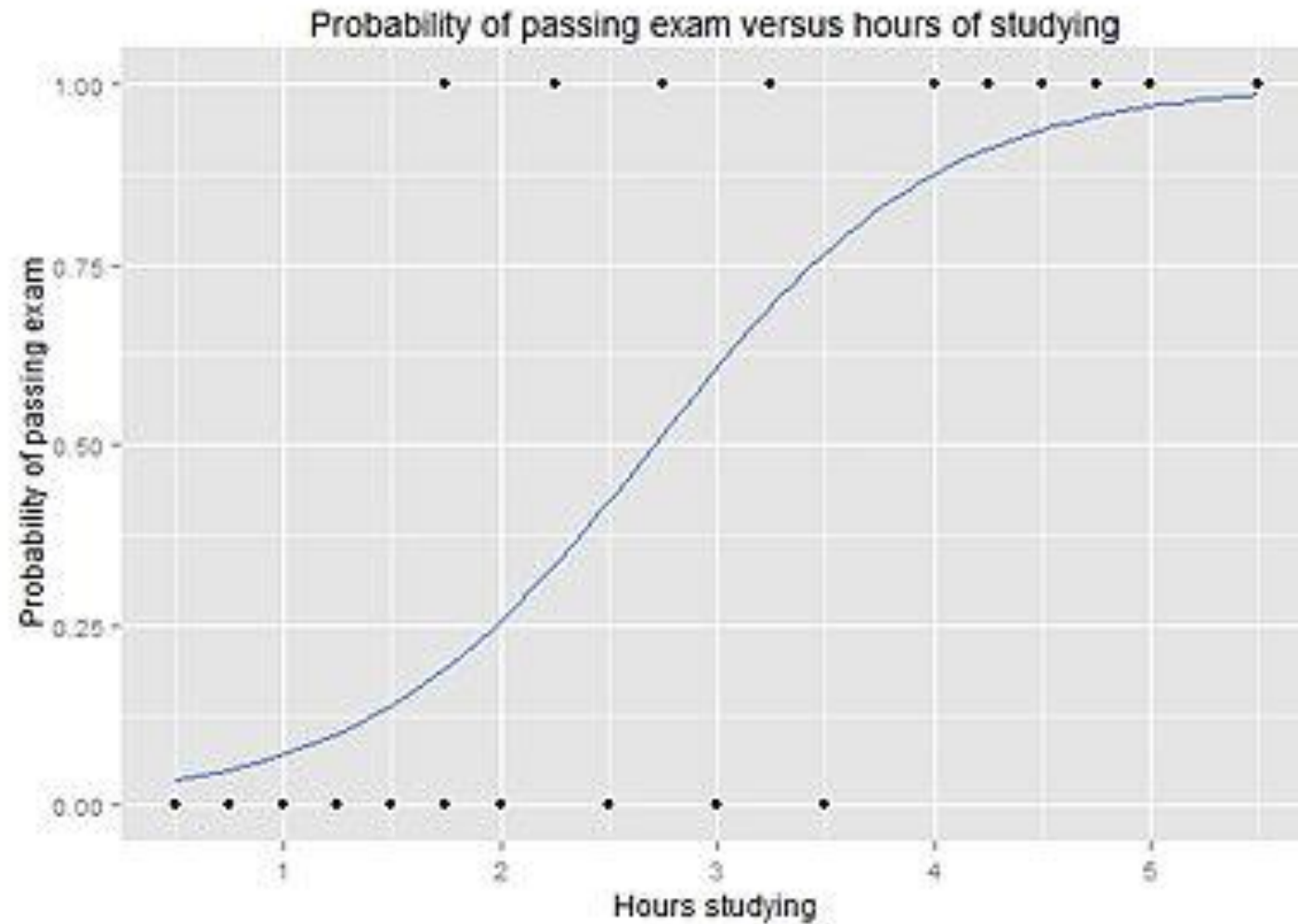
$$\text{Find } \min_{\alpha, \beta} Q(\alpha, \beta), \quad \text{for } Q(\alpha, \beta) = \sum_{i=1}^n \varepsilon_i^2 = \sum_{i=1}^n (y_i - \alpha - \beta x_i)^2$$

We won't worry about *how* we get the values here

Demonstrate on board

- Apply to a single X, Y
- What about when x is a vector of variables instead of a single number?

Logistic Regression

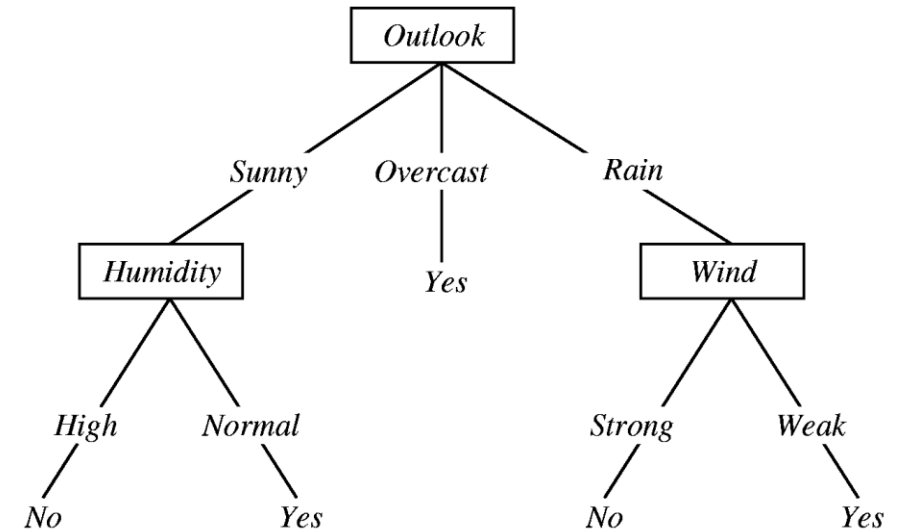


Linear Regression

- Pros:
 - Fast to build
 - Fast to use
 - Simple but powerful
- Cons:
 - Too simple for complex mappings
 - Overly dependent on training data
 - Can't work with partial information
 - Errors can massively harm its performance

Decision Trees

- Can be used for regression or classification tasks
- Learn a tree to map input to output from training data



Decision Tree Algorithm

```
set currentNode to head
  add all training data to head
openNodes = []
while currentNode.entropy > threshold:
  currentNode = openNodes.pop()
  children = calculate best split point(s)
  for child in children:
    if child.entropy <= threshold:
      make leaf node
    else:
      openNodes.push(child)
return head
```

*entropy is the measure of how much in agreement the data “in” this node is

Demonstrate on Board

Decision Trees are suited to problems where...

- Data is in the form X, Y pairs
- Y has discrete values it can be (yes/no or in a category)
- $X \rightarrow Y$ is not a simple linear function
- Training data may contain errors/noise
- Training data may contain some X values without all attributes

Decision Trees

- Pros:
 - Fast to use
 - Robust to errors
- Cons:
 - Slow to build
 - Depending on threshold for entropy, can **overfit** to training data

Overfitting

- In which a training method gets *too* good at predicting training data at the risk of making mistakes during testing.
- Child sees three dogs, all of different breeds, and is told that they are each a different breed but all dogs
- Child sees a cat and asks what breed of dog it is

Random Forest

- A random forest makes use of a designer-specified number of decision trees, each trained on a random subset of the training data
- During testing, each tree of the random forest votes, with the majority (or average or median) answer taken.

Question(s) 2

Besides churn, pick some event in general or for a particular game designers might want to be able to predict.

Which technique would you use? Why?

How could the designers use the learned model to improve the game?

Galactic Arms Race

