Interactive Bayesian Optimization for Game Mechanics

Abstract

fill me

Game design often involves a final phase of substantial fine-tuning of game assets. Paradigmatic examples include varying the settings of player-controlled character movement parameters, altering opponent combat statistics, or varying low-level parameters around movement and collision of game objects. Tuning is often a time-consuming and expensive process for several reasons:

1. parameter values must be set to (globally) optimal values, requiring search over a large space
2. evaluation of a setting cannot be done analytically or via simulation but requires costly (in terms of time and money) direct human evaluation
3. quality of a set of parameters may be difficult to specify on a global scale, but instead be relative to other sets of parameters

Interactive Bayesian optimization (closely related to active learning (Settles 2012) and sequential experimental design (Chaloner and Verdinelli 1995)) approaches can address these issues through optimization of design objectives that are expensive to evaluate (Brochu 2010). Employing non-parametric models (here Gaussian Processes) we demonstrate the application of interactive Bayesian optimization to two cases studies of game design tuning in a shoot-'em-up game: (1) optimizing player controls to player preferences and (2) adjusting enemy design parameters to enforce a desired level of player behavior.

For control optimization we demonstrate how a preference-learning approach can provide potential control settings to be tested and evaluated against the previous set of controls. Bayesian optimization affords automatic exploration-exploitation trade-offs that enable rapidly (globally) optimizing controls to player preferences via pairwise preference feedback. For enemy design optimization we demonstrate how a designer-specified objective function for player performance statistics can guide building a regression model from enemy parameter settings to desired design features.

First, we discuss related work in game tailoring and adaptation. Second, we motivate and describe our interactive Bayesian optization appraoch, detailing the Gaussian process regression and preference learning models. Third, we describe our shoot-'em-up game and describe two empirical human studies demonstrating the efficacy of our approch. We conclude by discussing extensions and the range of applications of this modeling approach.

Related Work

(Yu and Trawick 2011) uses SVM, hard to AL on this

(Hunicke and Chapman 2004) ad hoc

(Yannakakis and Hallam 2009) (Yannakakis, Maragoudakis, and Hallam 2009) GP pref model employed, not found best. however, we use more sophisticated kernel fn in order to better adjust fit parameters across dimensions and get back information on relative importance of dimensions. also note that GP pref generally has highest performance except in one case

(Bakkes, Spronck, and van Lankveld 2012)

Interactive Bayesian Optimization

Sequential Bayesian optimization is a modeling approach where a function is optimized through a sequence of points that are tested, each one selected by some algorithm based on previous points. Two functions are involved: (1) the *objective function* that maps inputs to outputs; and (2) the *acquisition function* that maps potential inputs to their value for optimizing the objective function output. In our application we employ Gaussian processes (GPs) for the objective function and a modified expected improvement acquisition function. Our approach falls under the umbrella of interactive Bayesian optimization as each point selected uses feedback provided in interaction with a human (Brochu 2010).

Gaussian processes are a widely used non-parametric modeling technique able to capture complex non-linear relationships within a data set, automatically adjust model complexity to data, and integrate out parameters without user intervention (Rasmussen and Williams 2006). Intuitively, non-parametric models are models that allow for an infinite number of variables to account for the data before selecting only the subset needed to explain a given set of observations. In practice, this leads to models that automatically become more complex to fit a dataset as needed. Bayesian formulations of GP regression and classification automatically trade-off between complexity of a model and fit to a data set, avoiding overfitting and poor generalization problems that occur with optimization approaches. Bayesian model specifications allow parameters of the model to be integrated out, simplifying their use by requiring less user specification. We employ GPs to leverage the benefits of: non-linear mapping from inputs to outputs, automatic complexity adjustment with data collection, and reduced or eliminated parameter specification from users.

Below we describe the formulation of GP regression and GP preference learning and then integrate GP models with active learning methods. Gaussian process regression enables automatic difficulty adjustment by modeling player performance in a game as a non-linear function of game parameters. Gaussian process preference learning enables optimization of game parameters (here controls) to player preferences by modeling player preferences for a set of game parameters as a non-linear function of game parameters than is then forced to pairwise choices between alternatives. Active learning uses a GP objective function to identify desirable next parameter settings to test, guided by a designer-specified acquisition function—here expected improvement—for parameter adjustment. For game performance, designers specify a goal of achieving a given level of in-game performance. For controls, designers specify optimal player preference.

Gaussian Process Regression

Gaussian processes are formally defined as “a collection of random variables, any finite number of which have a joint Gaussian distribution” (Rasmussen and Williams 2006). While allowing an infinite number of variables to be used, any GP model can be computed through a multivariate Gaussian distribution based on the input and output values. Gaussian processes are specified by their mean function ($m(x)$) and covariance function ($k(x, x')$):

$$f(x) \sim GP(m(x), k(x, x'))$$

Intuitively, GP regression learns a model predicting that similar inputs—according to the covariance function—should yield similar outputs. Different choices of the covariance function define different notions of similarity. In our work we employ the automatic relevancy detection (ARD) version of a squared exponential distance:

$$k(x, x') = exp\big(- \frac{1}{2} \Sigma_{l=1}^{d} \kappa^l (x^l - x'^l)^2 \big)$$

where $\kappa^l > 0$ is the ARD parameter for the l-th feature of a d-dimensional data set, serving to control the contribution of this feature to the model. Automatic relevancy detection allows us to optimize model parameters during the fitting process, automatically scaling input dimensions to minimize the impact of irrelevant aspects of the data. Mathematical properties of the GP mean that an initially zero valued mean function will taken on non-zero values after fitting data, allowing the model to be initialized with zero as the mean value (see (Rasmussen and Williams 2006) for additional details on GP regression). In our case we will use such a zero-mean GP.

For our performance regression model we predict player performance (number of times hit) from game parameters controlling enemy attacks (speed and size of bullets along with firing rate). We fit a GP regression model to player data and optimize the covariance function ARD parameters using stochastic gradient descent after each training point received. Since GP regression has a closed-form solution for learning and prediction this task can be done in near-real time with no appreciable time requirements ($<$ 1 second for

N training points

).

Gaussian Process Preference Learning

We employ a pairwise preference learning model rather than using preference rating scales due to human biases. Sequential numeric ratings are subject to a cognitive anchoring bias where earlier numeric ratings influence choices on subsequent ratings . We thus employ a model that generalizes information gained from pairwise rankings to the underlying preference of users for different instances (here game parameter settings). Games can only be played sequentially during comparisons, motivating an approach of pairwise preference ratings comparing each new instance to the previous one.

cite Kahn
man

Gaussian process preference learning models user choices in a two-part model: (1) a GP regression model specifying the underlying (unobserved) value of a single instance; (2) a probit model of how a choice is generated based on two instance being compared (Chu and Ghahramani 2005). The GP model allows a flexible specification of how users value a given instance specified in terms of its parameters. The probit model—known in economics as the Thurstone-Mosteller law of comparative judgment—converts a pair of latent values into a comparison judgment according to the function:

$$P(x_i \succ x_j | f(x_i), f(x_j) = \Phi\Big(\frac{f(x_j) - f_(x_j)}{\sqrt{2}\sigma_{noise}} \Big)$$

where x_i, x_j are two instances, $f(x_i)$ is the GP latent value of an instance, Φ is the cumulative normal distribution, and σ_{noise} is the inherent noisiness of comparative judgments. Intuitively, the probit model encodes preference judgments as based on the difference in underlying value of two instances, allowing for noise in preference ratings.

Due to the non-linear probit model used GP preference learning has no analytic learning model. Instead, we follow work by (Chu and Ghahramani 2005) and use a Laplace approximation to learn the underlying GP model's parameters. We employ a GP with zero mean and the ARD covariance function and optimize its parameters along with the selection of σ_{noise} using a grid search over the space of possible parameters . Unlike GP regression fitting, these nested optimization processes are computationally expensive, requiring . However, we note that optimization may be performed using any-time algorithms (such as DIRECT), allowing opti-

better op-
timization
method

X second
fit N train
points wit
dimension

jones et a
1993

mization of parameters to occur while the player plays a new option. In our experiments we chose to optimize parameter values and force players to wait in order to test the best-case performance of our approach.

Active Learning

Active learning (AL) is an approach to machine learning problems with a large set of unlabeled instances where a computer asks a human to provide information about given instances to learn a model of the instances as a whole (Settles 2012). AL is well suited to our application where the space of game parameterizations is very large and information can only be gained through the expensive process of having a human play and provide feedback about a game instance. Acquisition functions specify how a given AL algorithm weights potential instances to test based on a goal of optimizing the objective function. In our case, the GP regression model seeks to minimize the difference between desired and actual player performance and the GP preference model seeks to find the highest latent value instance.

Many possible acquisition functions exist, varying in how the functions balance the exploration-exploitation trade-off guiding how locally-tethered the search for large objective functions is (Settles 2012). Expected improvement (EI) is an acquisition function that balances the value of unseen instances against the uncertainty regarding their values. EI intergrates over all possible results to get an average-case estimate of the result, rather than seeking a best-case (or worst-case) scenario. We employ a modified EI function that incorporates a slack paramater ($\xi \geq 0$) to control the relative weighting of exploration and exploitation goals :

Lizotte

$$EI(x) = \begin{cases} (f(x) - f(x^+) - \xi\Phi(Z)+ & \\ \sigma(x)\phi(Z)) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases}$$

where $f(x)$ is the function value at x, x^+ is the instance with the current greatest function value, σx is the uncertainty in the value of the instance, $\phi(Z)$ is the Gaussian distribution density at Z and Z is defined as:

$$Z = \begin{cases} \frac{f(x)-f(x^+)-\xi}{\sigma(x)} & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases}$$

Intuitively, Z is the noise-scaled difference between the test point x and the current best point x^+, and the expected improvement takes a weighted combination of this gain against the uncertainty of the point. Points that are more uncertain and expected to have higher values are preferred to those with lower values or high values that are highly certain. ξ allows an explicit specification of how heavily to emphasize exploration

Experiment

We test our mechanics adaptation approach in a shoot-'em-up arcade game getting (Figure). An arcade-style game enables our system to use a series of waves to test game parameter settings and gather feedback from the player, iteratively refining the parameter settings to a given design objective. The fast, reflex-based gameplay style of the game emphasizes aspects of game "feel" and helps reduce complexities involved in long-term player planning and strategy.

In this section we describe our game domain and the two empirical human studies we conducted. The first study examines the use of GP regression to find and continually adjust game parameter settings to achieve a designer-specified level of performance. We show

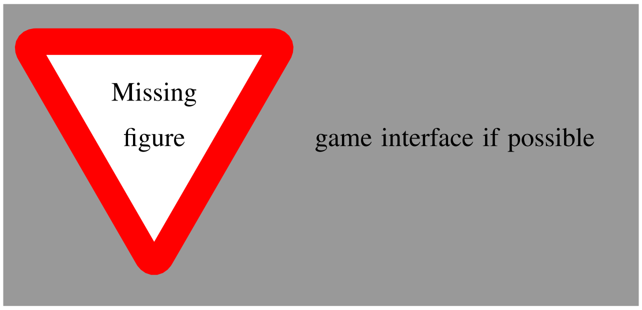
results: able to achieve small error; compare to other method?

. The second study examines the use of GP preference modeling to optimize controls to player preferences. We show

results: able to match player self-set values?; compare to other model?; show number iterations needed to succeed is small?

.

Game Domain



Shoot-'em-up games emphasize reflexes and pattern recognition abilities as a player maneuvers a ship to dodge enemy shots and return fire. In our game players are scored based on the number of times they are hit: players continually gain points over time and lose points when hit by opponents. Players are encouraged to avoid enemies as best as possible either through dodging enemy attacks or destroying enemy ships to reduce the number of incoming attacks.

update sc
ing as nee

Player controls are limited to shooting at enemies and moving their ship. Ship movement is governed by two controllable parameters: drag and thrust. Drag is the "friction" applied to a ship that decelerates the moving ship at a constant rate when it is moving—larger values cause the ship to stop drifting in motion sooner. Thrust is the "force" a player movement press applies to accelerate the ship—larger values cause the ship to move more rapidly when the player presses a key to move. Combinations of thrust a drag are easy to tune to rough ranges of playability. However, the precise values needed to ensure the player has the appropriate level of control are difficult to achieve as player movement depends on how enemies attack and individual player preferences for control sensitivity (much like mouse movement sensitivity). Our preference-learning experiment allowed the system to set drag and thrust for the player while attempting to maximize player preference for a set of controls.

During each wave of enemies a series of opponents are created that fire bullets at the player. Enemy parameters used in our study consist of the: size of enemy bullets, speed of enemy bullets, and rate that enemies fire bullets. Increasing bullet size requires the player to move more carefully to avoid bullets. Faster bullets require quicker player reflexes to dodge incoming fire. More rapid firing rates increase the volume of incoming fire. Together these three parameters govern how much players must move to dodge enemy attacks, in turn challenging player reflexes. As with controls, getting approximate settings for these parameters is easy, but fine-tuning them for a desired level of difficulty can be challenging. Our performance adaptation experiment allowed the system to set enemy bullet size, speed, and firing rate to minimize the difference between player performance and designer-specified target performance.

Methods

Our 1

Results

Discussion

Acknowledgments

References

[Bakkes, Spronck, and van Lankveld 2012] Bakkes, S. C.; Spronck, P. H.; and van Lankveld, G. 2012. Player Behavioural Modelling for Video Games. *Entertainment Computing* in press:1–9.

[Brochu 2010] Brochu, E. 2010. *Interactive Bayesian optimization: learning user preferences for graphics and animation*. Ph.D. Dissertation, University of British Columbia.

[Chaloner and Verdinelli 1995] Chaloner, K., and Verdinelli, I. 1995. Bayesian experimental design: A review. *Statistical Science* 10 (3):273–304.

[Chu and Ghahramani 2005] Chu, W., and Ghahramani, Z. 2005. Preference learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine learning*, 137–144. ACM.

[Hunicke and Chapman 2004] Hunicke, R., and Chapman, V. 2004. AI for dynamic difficulty adjustment in games. In *Proceedings of the AAAI Workshop on Challenges in Game Artificial Intelligence*.

[Rasmussen and Williams 2006] Rasmussen, C. E., and Williams, C. K. 2006. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA.

[Settles 2012] Settles, B. 2012. *Active learning*, volume 6. Morgan & Claypool Publishers.

[Yannakakis and Hallam 2009] Yannakakis, G. N., and Hallam, J. 2009. Real-time game adaptation for optimizing player satisfaction. *IEEE Transactions on Computational Intelligence and AI in Games* 1(2):121–133.

[Yannakakis, Maragoudakis, and Hallam 2009] Yannakakis, G. N.; Maragoudakis, M.; and Hallam, J. 2009. Preference learning for cognitive modeling: a case study on entertainment preferences. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* 39(6):1165–1175.

[Yu and Trawick 2011] Yu, H., and Trawick, T. 2011. Personalized procedural content generation to minimize frustration and boredom based on ranking algorithm. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*.