Towards a Fair 'n Square Aimbot – Using Mixtures of Experts to Learn Context Aware Weapon Handling

Christian Bauckhage
Centre for Vision Research
York University
4700 Keele Street, Toronto M3J 1P3, Canada
bauckhag@cs.yorku.ca

Christian Thurau
Applied Computer Science
Bielefeld University
P.O. Box 100131, 33501 Bielefeld, Germany
cthurau@techfak.uni-bielefeld.de

ABSTRACT

Superior weapon handling is the road to success in first person shooter games. On the expert level of gameplay, this demands more than just accurate tracking and targeting of opponents. It also requires knowledge about the behavior and characteristics of different weapons in the player's arsenal. Given enough practice, human players usually master both skills. Usual gamebots, in contrast, can only track and target but lack the sense for the appropriateness of a weapon in a given situation. As a consequence, their performance and behavior appears non-natural. In this paper, we propose to make use of machine learning techniques to realize aiming behavior that is adapted to the spatio-temporal context. We will present a mixture of experts architecture of neural networks which is specialized in the use of several weapons. Experimental result show that this paradigm indeed is able to produce more human-like aiming behavior.

Motivation and Background

The genre of first person shooter games is arguably among the most popular computer game genres[1]. In a FPS game, the player moves through a virtual world (also called a *map*) which he perceives from the first person perspective. Though several tactical variations exist, his main task basically is to battle against other characters on the map.

Virtual combat is carried out by means of weaponry that varies from game to game. However, something all FPS games have in common is that the provided weapons exhibit different characteristics. A typical virtual firearm might be devastating but take long to recharge, another might deliver a high frequent hail of bullets each of which will only have a rather low impact, or there might be an incarnation of a sniper rifle suited for long distance engagement but of minor use for close combat.

As these examples indicate, success in an FPS game may require more than virtual amok run. And indeed, observing professional or semi professional cyber athletes play reveals that they chose their arms according to the game state they encounter. For example, experienced players switch to a lower impact weapon after having hit the opponent with a high impact one. They know that if the opponent survived the first blow, firing another valuable high impact bullet would be a waste. A quick strike with a weapon of lesser strength will also yield the point.

In contrast to human players, artificial agents (also called *gamebots*) lack this level of sophistication. Since their behavior is still largely rule based, the set of variables that determine their weapon handling is comparatively small. While human players typically (and often unconsciously) chose their actions according to a whole zoo of parameters (e.g. the distance to an opponent, the local topography of the map, the current amount of ammunition, or known advantages and disadvantages of a weapon), gamebots often are programmed to have a weapon of choice and to use that weapon regardless of the current context of the game. In order to nevertheless provide a challenge on higher levels of a game, programmers compensate for the lack of bot intelligence by means of supernatural, unerring aiming: the bot reads the opponent's location from an internal variable and therefor simply cannot miss the target.

In this contribution, we will report on an avenue towards more situation awareness for gamebots. Considering the game QUAKE II® by ID software as a practical example, we will follow an idea proposed in an earlier contribution (Bauckhage, Thurau & Sagerer 2003). We will apply neural network architectures to analyze and learn from the data encoded in the network traffic produced by human players. In particular, we will discuss the idea of using a *mixture of experts* to obtain a collection of neural networks that are specialized in the use of several weapon. We shall present experiments which indicate that such an architecture can act as a context sensitive weapon selection mechanism. First, however, we will roughly survey related work in machine learning for commercial multiplayer computer games as well as briefly summerize the theory behind the mixture of experts concept. A summary and an outlook will close this contribution.

[1]According to the game portal Gamespy (08, 2004), 18 of the 20 most popular online games belong to the FPS category; in August 2004, there were more than 60.000 servers with more than 150.000 players online every minute.

Related Work

Currently, we are witnessing an increased academic interest in machine learning for the design of believable computer game characters. One of the boosting factors behind this boom can be concluded from observations reported by authors such as Cass (2002) or Nareyek (2004): on the one hand, up to today, most commercially available games rely on deliberative AI techniques like finite state machines or A^* searches. On the other hand, subsymbolic machine learning as a tool to produce life-like game agents has been largely neglected by the scientific community. However, this situation is about to change.

Recent work by Spronck, Sprinkhuizen-Kuyper & Postma (2003) introduced reinforcement learning to the task of rule selection for agent behavior in a commercially available role playing game. Earlier, the same authors reported on a hybrid coupling of genetic algorithms and neural networks for offline learning in a strategy game (Spronck, Sprinkhuizen-Kuyper & Postma 2002).

Neural networks were also reported to perform satisfiable in the genre of first person shooter games. Given the data contained in the network traffic of a multiplayer game, different architectures of neural networks were applied to learn life-like behavior for artifical characters. This was accomplished on the level of mere *reactive* behavior (Bauckhage et al. 2003, Thurau, Bauckhage & Sagerer 2003) as well as for *strategic* behaviors like goal oriented path computation (Thurau, Bauckhage & Sagerer 2004*a*). More recent work based on biologically inspired modeling and clustering techniques (movement primitives) even indicates that reactive and strategic components of behavior can be generated using a unified framework (Thurau, Bauckhage & Sagerer 2004*b*). All these approaches produce believable behavior for computer game characters. However, subsymbolic solutions for the *tactical* level of gameplay are still missing. According to a widely accepted psychological hierarchy of human behavior (Hollnagel 1994), tactical decisions require more situation awareness than reactive behavior but demand less long-term planning than a strategy. In the following, we will argue that multi-classifier systems like for example mixture of experts architectures may provide an appropriate answer to this problem.

Mixtures of Experts

Given a set of pairs of vectors $\{(\vec{x}^\alpha, \vec{y}^\alpha)\}$ where $\vec{y}^\alpha = f^*(\vec{x}^\alpha)$, a multi layer perceptron can learn a function f that approximates f^* by minimizing the sum of errors

$$E(\mathbf{W}) = \frac{1}{2}\sum_\alpha \left(\vec{y}^\alpha - f(\vec{x}^\alpha, \mathbf{W})\right)^2$$

where $\mathbf{W}$ is the weight matrix of the network. For a Gaussian error $\left(f(\vec{x}^\alpha, \mathbf{W}) - \vec{y}^\alpha\right)$ this approach will produce the maximum likelihood solution for the parameters w_{ij}. Alas, a single Gaussian mode is an unrealistic assumption for most

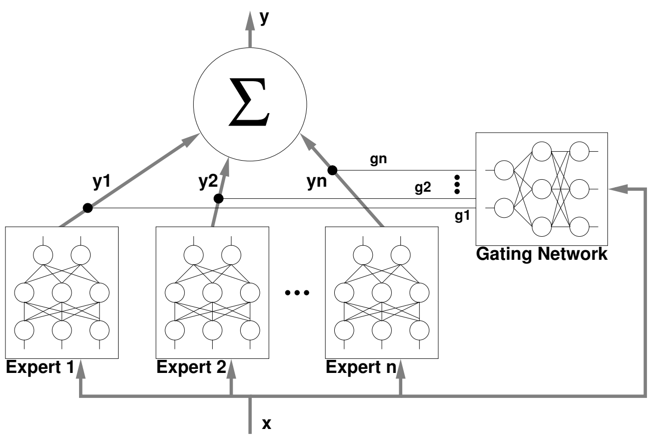


Figure 1: A mixture of experts architecture consists of a set of expert networks and a gating network. Given an input $\vec{x}$, the output $\vec{y}$ is computed as the sum $\vec{y} = \sum_j g_j(\vec{x}) y_j(\vec{x})$.

real data. The mixture of experts (MOE) approach as introduced by Jacobs, Jordan, Nowlan & Hinton (1991) tackles this problem by means of a divide and conquer strategy. Instead of fitting a global function into the training data, it uses a mixture of local experts which are moderated by a gating network. In the following, we will briefly summerize this technique. Readers familiar with the topic should skip to the next section.

As shown in Fig. 1, the basic idea of the MOE approach is to compute the vector $\vec{y}$ as a weighted sum of outputs produced by n expert networks. Given a vector $\vec{x}$, the corresponding $\vec{y}$ thus results from

$$\vec{y} = F(\vec{x}, \theta) = \sum_{j=1}^{n} g_j(\vec{x}, \mathbf{V}) f_j(\vec{x}, \mathbf{W}_j)$$

Note that $\theta = (\mathbf{V}, \mathbf{W}_1, \ldots, \mathbf{W}_n)$ denotes the set of all model parameters. It is common to require $\sum_j g_j = 1$ which can be realized by defining the g_j to be soft-max functions of the output layer values s_j of the gating network:

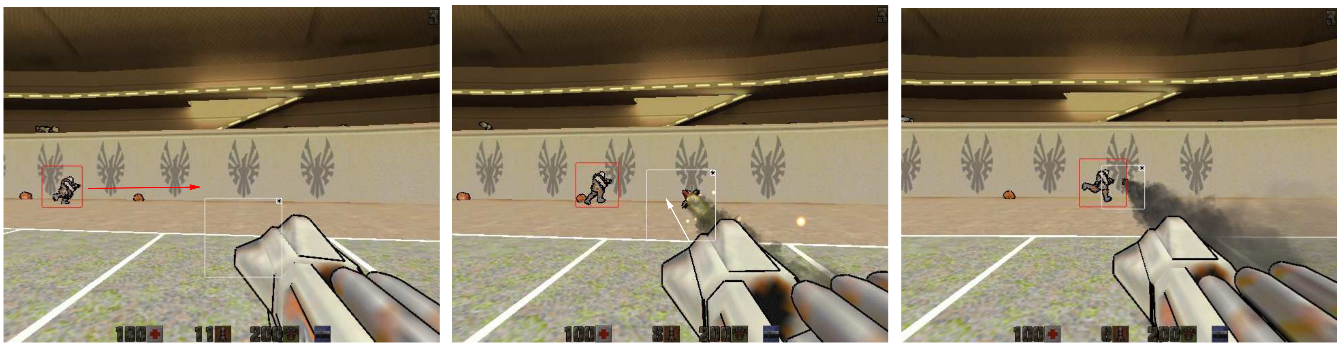
$$g_j(\vec{x}, \mathbf{V}) = \frac{e^{s_j(\vec{x}, \mathbf{V})}}{\sum_k e^{s_k(\vec{x}, \mathbf{V})}}$$

This quite naturally leads to a probability interpretation of the weights g_j.

The naive approach to the estimation of the parameters θ would be to minimize the error

$$E(\theta) = \frac{1}{2}\sum_\alpha \left(\vec{y}^\alpha - F(\vec{x}^\alpha, \theta)\right)^2$$

by a gradient descend $\nabla_\theta E$. But the MOE model allows for a more elegant solution. Consider the following statistical interpretation: each pair $(\vec{x}, \vec{y})$ is generated by a random process that first chooses $\vec{x}$ according to a given density and then randomly selects an expert according to the probability $g_j(\vec{x})$. The chosen expert j generates a random variable whose mean


(a) Targeting its expected position . . .

(b) . . . an sending the missile there . . .

(c) . . . will most likely hit the opponent.

Figure 2: Example of experienced handling of the QUAKE II® rocket launcher. In contrast to the 'direct hit' weapons in the arsenal, rockets are considerably slow. This requires to anticipate the opponent's movement and to target its expected rather than its current position. A gamebot will have to reproduce this behavior in order to appear life-like.

is $\vec{y}_j = f_j(\vec{x}, \mathbf{W}_j)$. The vector $\vec{y}$ then is the expected value $\mathbb{E}(\vec{y}_j|\vec{x})$. The probability of a pair $(\vec{x},\vec{y})$ can hence be modeled as

$$\begin{aligned} P(\vec{x},\vec{y}|\theta) &= \sum_j g_j(\vec{x},\mathbf{V}) P(j,\vec{x},\vec{y}|\mathbf{W}) \\ &= \sum_j g_j(\vec{x},\mathbf{V}) \mathcal{N}_j e^{-\frac{1}{2\sigma_j^2}\left(\vec{y}-f_j(\vec{x},\mathbf{W}_j)\right)^2} \end{aligned}$$

where $\mathcal{N}_j$ is a normalization factor. Assuming independence of the pairs in the set $\{(\vec{x}^\alpha,\vec{y}^\alpha)\}$, the set's log-likelihood is thus

$$\begin{aligned} L(\theta) &= \log \prod_\alpha P(\vec{x}^\alpha,\vec{y}^\alpha|\theta) \\ &= \sum_\alpha \log \sum_j g_j(\vec{x}^\alpha,\mathbf{V}) \mathcal{N}_j e^{-\frac{1}{2\sigma_j^2}\left(\vec{y}^\alpha-f_j(\vec{x}^\alpha,\mathbf{W}_j)\right)^2} \end{aligned}$$

This likelihood approach is advantageous because now the familiar expectation maximization algorithm can be used to estimate the optimal set of parameters θ^*. But even a gradient descent $-\nabla_\theta L$ will converge faster and more reliable than in the case of $E(\theta)$ for the surface of $-L(\theta)$ is smoother and comes along with less local minima as Rao, Miller, Rose & Gersho (1997) point out.

Note that a MOE model simultaneously learns how to segment the input space and how to model the mappings from the resulting segments to the output space. In contrast to cluster algorithms like, for instance, k-means, the MOE approach yields a soft partition of the input space. Finally, extensions to hierarchical MOE models are possible where each expert itself represents a mixture of experts.

Experiments

In a first series of experiments, we examined whether a mixture of experts architecture can learn human-like handling of different fire arms in QUAKE II®. We confined this examination to a subset of three weapons and, for the time being, considered the *blaster*, the *rocket launcher* and the *railgun*. Roughly speaking, the railgun is a powerful long-distance weapon with considerable recharge delay. The rocket launcher is a devastating tool in mid-distance combat. Human players tend to avoid its use in close combat because the splash of rockets can cause self harm. The blaster is a handgun with a high bullet frequency suited for close combat where it is most likely to harm opponents.

According to their characteristics, these weapons are typically used in different (spatial) contexts. However, their use does not only differ with respect to the distance to an enemy player; their ballistics vary as well. While the railgun and the blaster are instant hit weapons, the missiles fired by the rocket launcher are rather slow. This requires to add anticipation to the handling of the rocket launcher. As it is exemplified in Fig. 2, instead of directly targeting their opponents, experienced players fire rockets towards a location where their adversaries will most likely appear in a couple of moments.

Given these observations, the question is thus if a multi-classifier architecture can learn to select a weapon according to the distance to an opponent and simultaneously learn to produce appropriate aiming.

Setup

In order to investigate this question, we recorded several demos. In each of these, two players met on a custom map called *stadium*; their use of the three different weapons corresponded to the above characteristics.

situation		# training vectors	# test vectors
blaster	aim	2079	693
	shoot	1572	525
rocket	aim	4104	1369
	shoot	1537	513
railgun	aim	6508	2170
	shoot	1870	624

Table 1: Data used in experiments.

training method	pre-trained experts	# hidden neurons	E_{TRAIN}	E_{TEST}
$\nabla_\theta L$	no	4	0.01318	0.01280
EM	no	4	0.01332	0.01293
EM	no	7	0.01125	0.01133
EM	yes	7	0.01153	0.01167

Table 2: Experimental results.

The data that was extracted from this demos in order to train the classifier consisted of pairs $\{\vec{x}^\alpha, \vec{y}^\alpha\}$ at a frequency of 10Hz where the input vectors $\vec{x}^\alpha \in \mathbb{R}^5$ encoded the spatial angle and the distance between the player and its opponent as well as the yaw and pitch angle of the player. The binary output vectors $\vec{y} \in [0,1]^4$ consisted of flags indicating the type of weapon to be used and a flag indicating whether to shoot or not.

Three quarters of the recorded material were used for training, the remaining quarter was for testing. Table 1 summerizes how many vectors were given for each weapon and how many of them corresponded to actual shooting behavior. Note that these figures reflect the recharging times of the considered weaponry. While for the blaster the ratio of shots to observations is approximately $3/4$, for the railgun it is about a mere $2/7$.

We experimented with a MOE architecture of three experts and a gating network. The gate was chosen to have 10 hidden neurons, the experts were tested with 4 and 7 hidden neurons, respectively. Training was done in 5000 epochs using gradient descend or the EM algorithm and applied the Super-SAB method for learning rate control (cf. e.g. (Sarkar 1995)). For the sake of comparison, we also tested an architecture where the three expert networks were pre-trained, one for each weapon, and the gate was subsequently trained to select the most appropriate expert.

Results

All configurations tested in our experiments were able to reproduce the desired behavior. An in-game examination of the resulting network architectures showed bots that switched between the considered weapons according to the handling encoded in the training data. If the opponent was close, the bots selected the blaster, in mid-distance combat they referred to the rocket launcher and for long-distance fights they used the railgun. Also, the bots reproduced the aiming behavior contained in the demos. When using the blaster or the railgun they directly targeted their enemies; when using the rocket launcher they 'predicted' their adversaries trajectory and fired accordingly. Finally, the shooting frequencies characteristic for the different weapons were learned as well.

Table 2 displays some of the figures we obtained from an offline evaluation. Obviously, the architectures with expert networks with 7 hidden neurons perform better even though, compared to the ones with 4 hidden neurons, the gain is small. What is noticeable, is that the difference between training error and test error is small in every case; this backs the above observation of a good reproduction of the behavior encoded in the training material.

In one of the cases considered in our experiments, training via gradient descend produced a slightly better result than the EM parameter optimization. Pre-trained experts performed generally slightly worse than those trained together with the gating network (s. last row of Tab. 2).

To our surprise, none of our experiments resulted in an architecture where there was a clear assignment of the three expert networks to the three different weapons. Rather, our experiment produced a holistic representation of the demo behavior distributed over the three networks.

Discussion

Mixtures of experts provide a means to learn the handling of different virtual weapons from human-generated training data. Simple combinations of small multilayer perceptrons (i.e. 5-4-4 or 5-7-4 network topologies) can learn to switch between weapons if the distance to an opponents changes correspondingly. Simultaneously, they can learn to produce appropriate aiming behavior, that is, to directly target the opponent with an instant hit weapon or to target the expected position of the enemy when using the rocket launcher.

These results are encouraging and hint that MOE approaches may also be able to treat more complex context dependent behaviors. However, these findings come along with a surprise. In none of our experiment did experts for the handling of a single weapon evolve. Instead, the gating networks resulting from the different configurations that were tested always generated soft combinations of the responses of the expert networks.

An explanation for this phenomenon could be that in the data space the handling of the different weapons does not differ as much as it appears from the in-game perspective. An effect like this was already encountered in a different context where we experimented with biologically inspired techniques to learn human-like movement skills (Thurau et al. 2004*b*). If this should be the case in our current experiments as well, the architectures might have learned the principal components or generalized principle components within the abstract data space and produce their output as a linear combination thereof. (Work by Fancourt & Principe (1997) revealed that MOE architectures can accomplish this.) Currently, we are conducting further experiments to verify this assumption or to gain more insight in what else is happening here.

Summary and Outlook

In this contribution we proposed to apply multi-classifier systems to learn different context dependent behaviors for computer game characters. The mixture of experts model provides an approach to the simultaneous training of a set of competing neural networks. Moderated by a gating network, subnetworks will emerge that are specialized in handling different regions of the data space.

For our experiments we used a MOE architecture of three multilayer perceptrons and a nonlinear gating network. Given training data that exemplified the use of three different weapons in the game QUAKE II®, the coupling of the networks indeed resulted in a system with expertise in using the three weapons. As a whole, the architecture learned in which context the considered weapons were most frequently used. It was learned that the *blaster* is usually invoked in situation of close combat where it is fired frequently. Concerning the use of the *rocket launcher*, the architecture produces a distance dependent offset in aiming on an opponent that compensates for the low speed of the missiles. For the *railgun*, in contrast, it learned to directly target opponents in farther distances. Even simple architectures of simple neural networks can thus reproduce human weapon handling. Therefor, the resulting gamebots do not have to rely on superhuman aiming in order to produce engaging gameplay; they are *fair* opponents.

Future work will have to consider larger expert architectures and more complex activities than weapon handling. This will have to include variables like *health* and *armor* which describe the internal state of a game agent and are responsible for more sophisticated tactics and behaviors. It will be interesting to see to what extend the technique applied in this contribution will scale. General experience in neural network research suggest that there might be a boundary. A remedy could be to extend the concept of movement primitive as described in (Thurau et al. 2004*b*) to the idea of *tactics primitives* which may be unified into an architecture similar to the mixture of experts approach.

Acknowledgments

This work was supported by the German Research Foundation (DFG) within the graduate program “Strategies & Optimization of Behavior”. We would like to thank our students Marion Wadermann, Klaus Schapp, and Matthias Steinrücken as well as all other participants of the seminar *Neural Quakebots* held at Bielefeld University in the summer term of 2004. Without their interest, diligence, and enthusiasm this contribution would not have been possible. Also, we are deeply grateful for the help of Dirk Raufer. The in-game illustrations in this paper were generated using a visualization tool he developped in his BS. thesis completed at Bielefeld University in 2004.

REFERENCES

Bauckhage, C., Thurau, C. & Sagerer, G. (2003), Learning Human-like Opponent Behavior for Interactive Computer Games, *in* B. Michaelis & G. Krell, eds, ‘Pattern Recognition’, Vol. 2781 of *LNCS*, Springer, pp. 148–155.

Cass, S. (2002), ‘Mind Games’, *IEEE Spectrum* pp. 40–44.

Fancourt, C. & Principe, J. (1997), Soft Competitive Principal Component Analysis Using the Mixture of Experts, *in* ‘DARPA Image Understanding Workshop’, pp. 1071–1075.

Gamespy (08, 2004), http://archive.gamespy.com/stats/index.shtm.

Hollnagel, E. (1994), *Human Reliability Analysis: Context & Control*, Academic Press.

Jacobs, R., Jordan, M., Nowlan, S. & Hinton, G. (1991), ‘Adaptive Mixture of Local Experts’, *Neural Computation* **3**(1), 79–87.

Nareyek, A. (2004), ‘Artificial Intelligence in Computer Games – State of the Art and Future Directions’, *ACM Queue* **1**(10), 58–65.

Rao, A., Miller, D., Rose, K. & Gersho, A. (1997), ‘Mixture of Experts Regression Modeling by Deterministic Annealing’, *IEEE Trans. on Signal Processing* **45**(11), 2811–2820.

Sarkar, D. (1995), ‘Methods to Speed Up Error Backpropagation Learning Algorithm’, *ACM Computing Surveys* **27**(4), 519–542.

Spronck, P., Sprinkhuizen-Kuyper, I. & Postma, E. (2002), Improving Opponent Intelligence through Machine Learning, *in* ‘Proc. GAME-ON’, pp. 94–98.

Spronck, P., Sprinkhuizen-Kuyper, I. & Postma, E. (2003), Online Adaptation of Game Opponent AI in Simulation and in Practice, *in* ‘Proc. GAME-ON’, pp. 93–100.

Thurau, C., Bauckhage, C. & Sagerer, G. (2003), Combining Self Organizing Maps and Multilayer Perceptrons to Learn Bot-Behavior for a Commercial Computer Game, *in* ‘Proc. GAME-ON’, pp. 119–123.

Thurau, C., Bauckhage, C. & Sagerer, G. (2004*a*), Learning Human-Like Movement Behavior for Computer Games, *in* ‘Proc. Int. Conf. on the Simulation of Adaptive Behavior’.

Thurau, C., Bauckhage, C. & Sagerer, G. (2004*b*), Synthesizing Movements for Computer Game Characters, *in* C. Rasmussen, H. Bülthoff, M. Giese & B. Schölkopf, eds, ‘Pattern Recognition’, Vol. 3175 of *LNCS*, Springer, pp. 179–186.