Tema 01. Introducción al aprendizaje Automático

Autor: Ismael Sagredo Olivenza

1. Qué es el aprendizaje automático (Machine Learning)

Learning

- “The act, process or experience of acquiring knowledge or Skills”

Automatic

- That works by itself

1.1 Machine learning definition

Arthur Samuel (1959):
"Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed"

Tom Mitchell (1998)
"Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some **task** (T) and some **performance** measure (P), if its performance on T, as measured by P, improves with **experience** E"

Tom M. Mitchell; Machine Learning; McGraw-Hill, 1998

Example

Handwriting recognition

– T: Recognise handwritten words given as bitmaps

– P: Porcentage of identified words

– E: A collection of bitmaps together with the words they represent

Autonomous driving

– T: driving on a motorway using vision sensors.

– P: distance travelled before making a mistake

– E: sequences of images together with the steering wheel movements

1.2 How do machines learn?

It is not yet clear how humans (or other animals) learn.

Learning: the ability of living things to improve their skills.

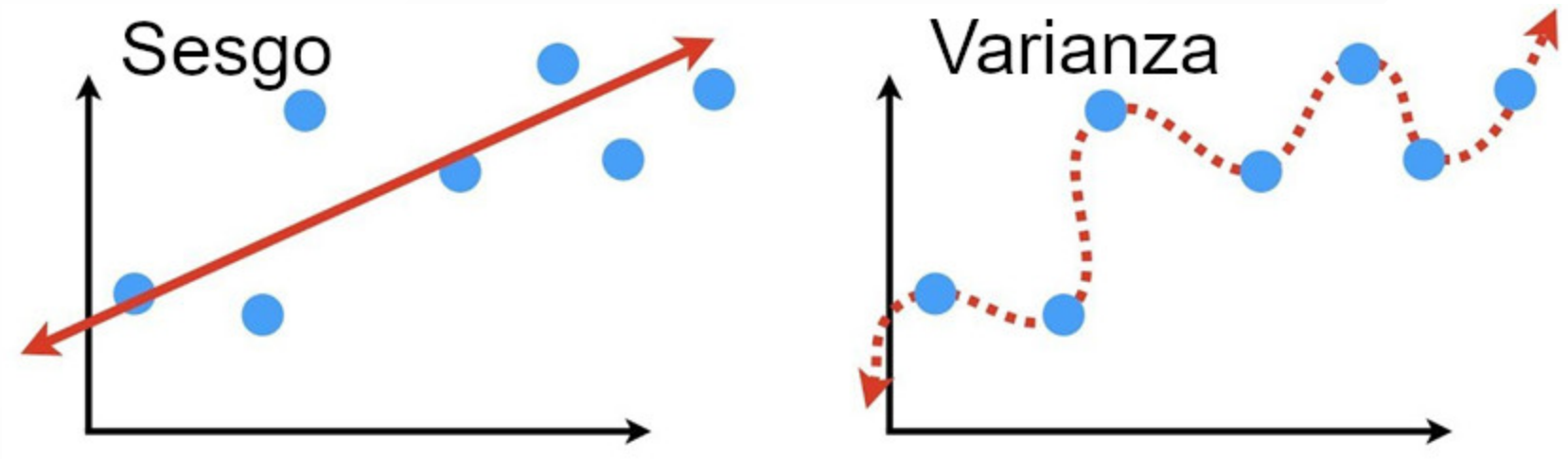
It requires **memory** to store new knowledge.

Poblem abstraction is required to solve new problems.

- Machine learning performs **inductive learning**, i.e. it draws conclusions based on examples.
- It requires **abstraction** from small details that do not provide information relevant to general cases.
- Abstraction generates **biases** (**Sesgos**) and these are necessary for learning work properly. This is known as **having generalizability** (**Capacidad de generalización**)

Do humans introduce biases into our learning?

- Spelling rules (some of which have exceptions)
- Electric equations assume that bodies are in vacuum
- Uniformly accelerated motion assumes you do it in a vacuum
- Einstein's equation of relativity (e=mc^2), only applies to particles with mass (for the photon the calculations are more complex).



Source https://skillsfuturetv.com/wp-content/uploads/2020/05/maxresdefault-360.jpg

Sesgo (Bias): trying to extract the general characteristics that describe them

Varianza: do not introduce bias (**Overfiting** / **sobre adaptación**)

1.3 Types of machine learning

- **Supervised Learning**: the examples that are introduced to learn also have the solution to the problem.
- **Unsupervised learning**: is carried out on examples that do not inform about the expected outcome.
- **Reinforcement Learning**: feedback is obtained from the outside world (context, environment) and not from examples.

Types of problems that can be solved

- **Classification**: identify to which of a set of categories a new observation belongs. Discrete output is expected
- **Regression**: The output is continuous. An infinite number of cases => estimate the output as accurately as possible.
- **Clustering**: The number and type of classes is not known a priori and we aim to discover them.
- **Dimension reduction**: To extract the relevant characteristics from the data.
- **Association**: Aims to establish relationships between variables (correlations).

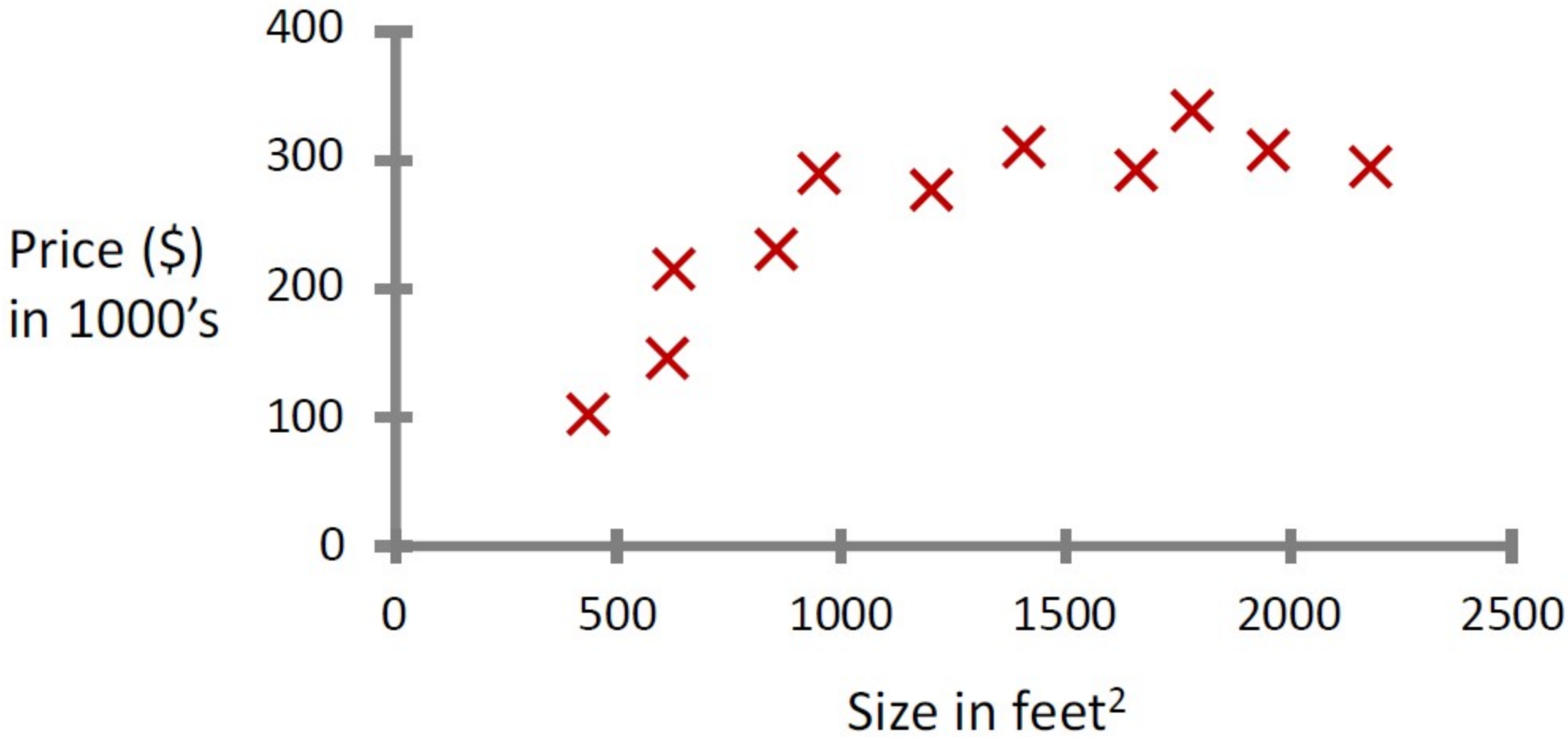
1.4 Supervised learning

Learns from being given the **right answers**

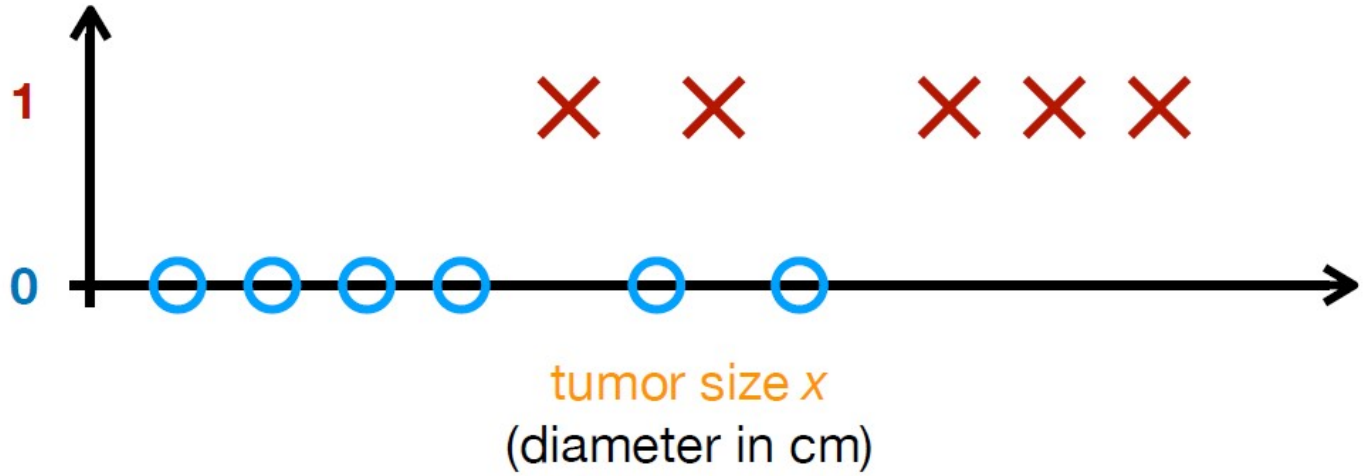
Weight	Height	Years	Years study	=	salary in thousands
20	170	30	10		**30**
27	175	35	5		**25**

- Example: Vector of N elements. Each element represent an attribute of the instancia
- Input: A collection of M examples. Matrix (MxN)
- Output: Vector of M solutions.

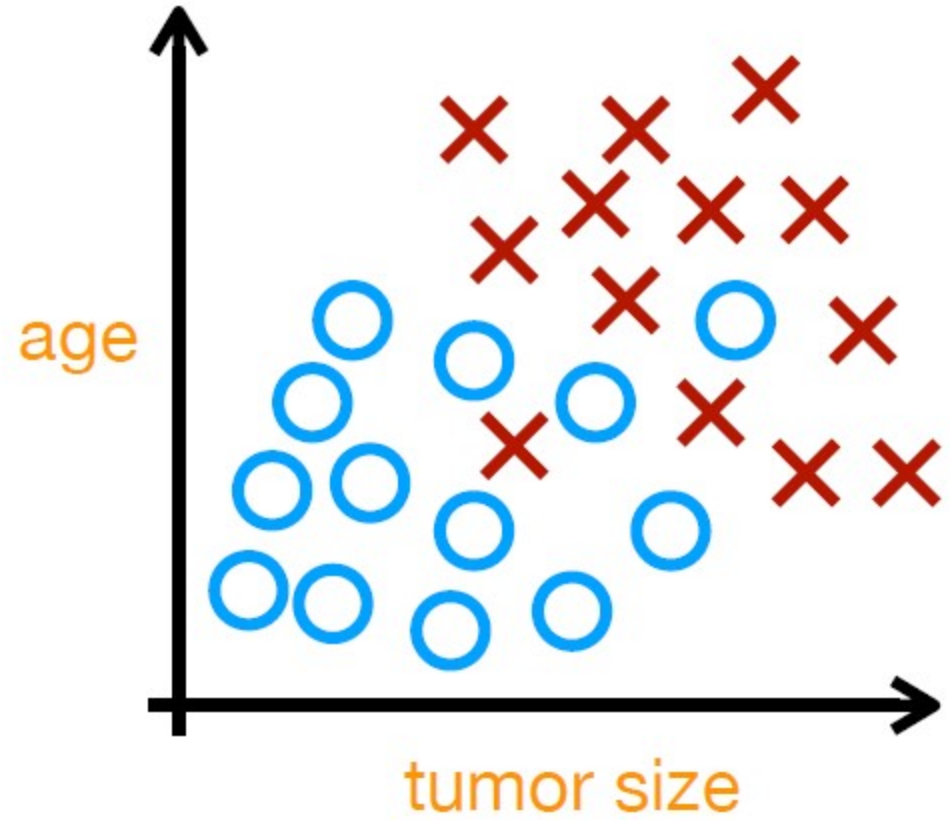
Regression: Housing price prediction



Classification: Breast cancer detection



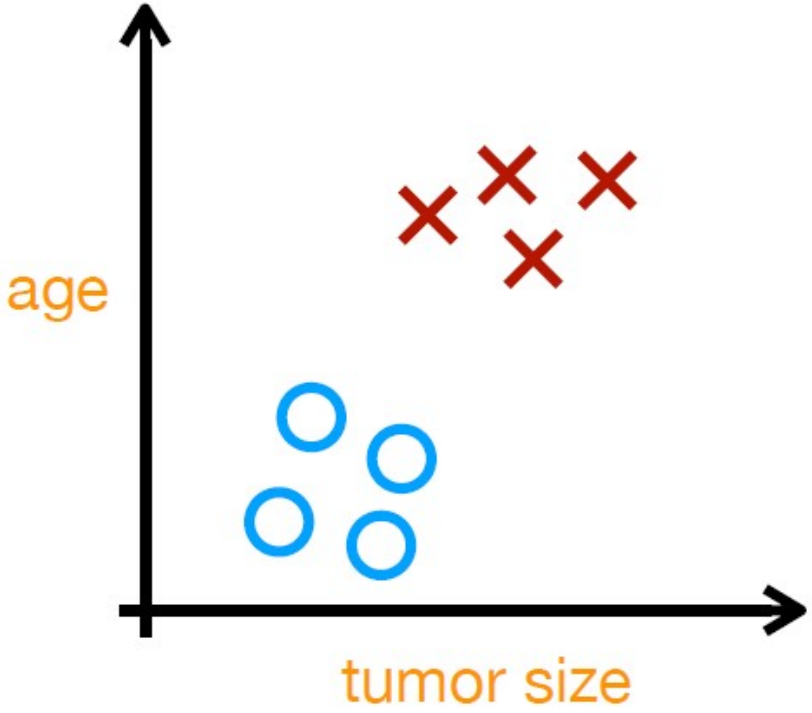
Two or more inputs



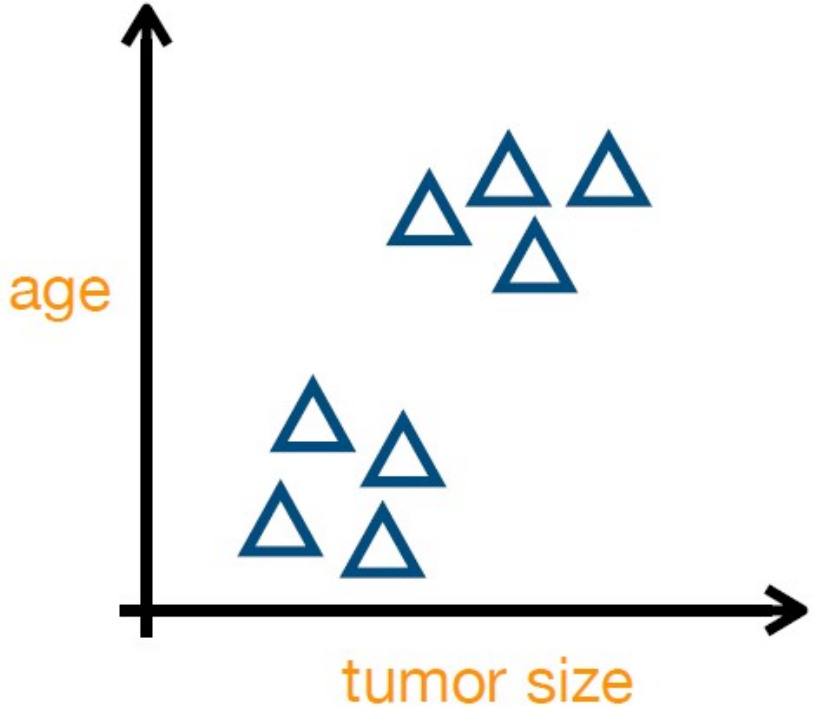
1.5 Unsupervised learning

- If I don't know the value we are trying to predict in the examples
- Extract the most relevant characteristics from a dataset
- We usually need a human interpretation to narrow down the meaning of the results obtained.

Supervised learning
Learn from data labeled with the "right answers"



Unsupervised learning
Find something interesting in unlabeled data



Unsupervised learning

Data only comes with inputs x, but not output labels y.
Algorithm has to find structure in the data.

Clustering
Group similar data points together.

Dimensionality reduction
Compress data using fewer numbers.

Anomaly detection
Find unusual data points.

1.6 Reinforcement learning

- System learns in online mode (not based on example)
- Environment must indicate to system if the actions are correct or not.
- Learn by trial and error
- Specially adapted to dynamic environments. The system will retrain itself if the environment changes.

1.7 Examples of Machine Learning in Videogames

- Designers don't want AI to hallucinate and generate emergent behaviors
- But its use is being actively researched.
- https://www.ea.com/technology/research/ai-machine-learning

Multi-Critic Actor Learning: Teaching RL Policies to Act with Style

Siddharth Mysore, George Cheng, Yunqi Zhao, Kate Saenko, Meng Wu

Published: 28 Jan 2022, Last Modified: 14 Feb 2023 ICLR 2022 Poster Readers: Everyone Show Bibtex Show Revisions

Keywords: Reinforcement Learning, Multi-Style Learning, Multi-Task Learning, Actor-Critic

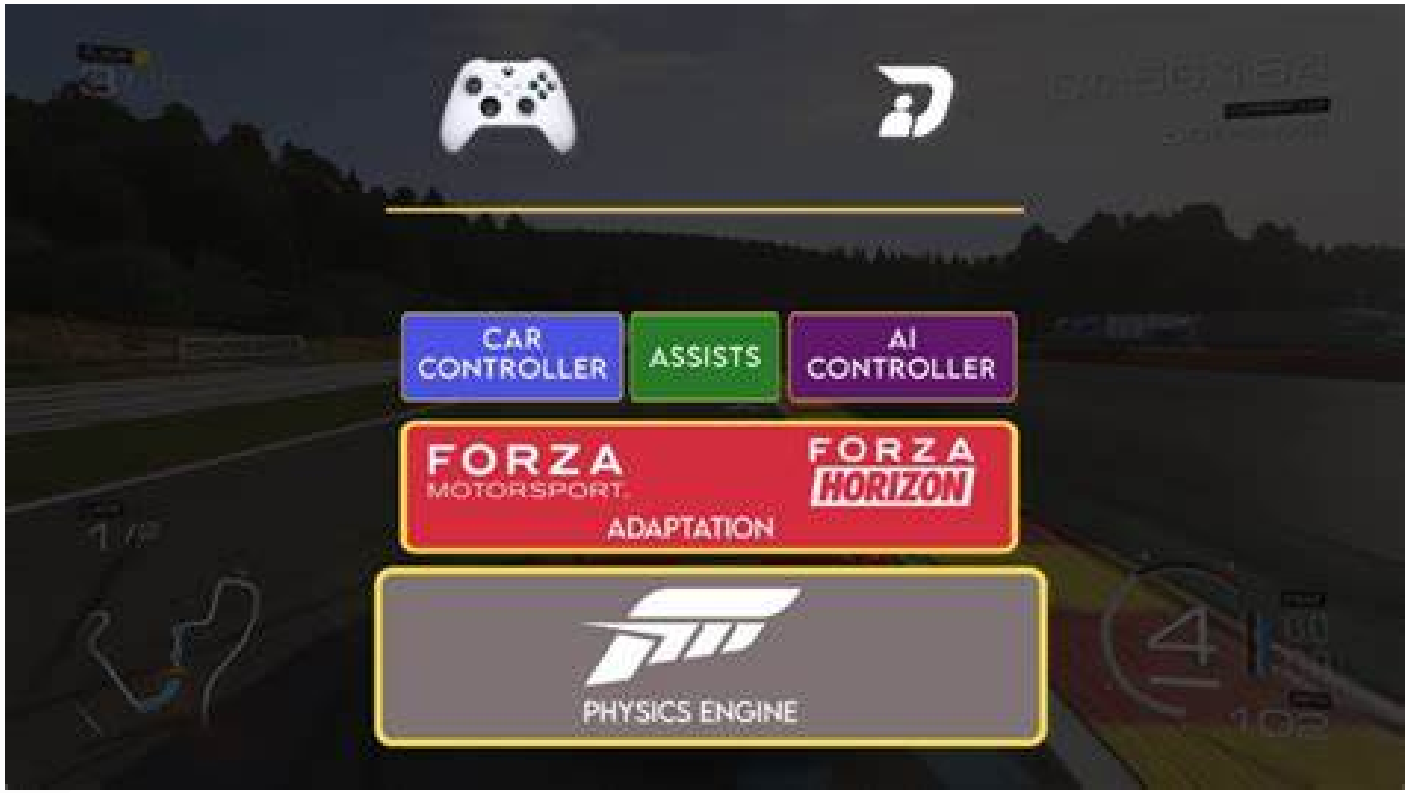
Abstract: Using a single value function (critic) shared over multiple tasks in Actor-Critic multi-task reinforcement learning (MTRL) can result in negative interference between tasks, which can compromise learning performance. Multi-Critic Actor Learning (MultiCriticAL) proposes instead maintaining separate critics for each task being trained while training a single multi-task actor. Explicitly distinguishing between tasks also eliminates the need for critics to learn to do so and mitigates interference between task-value estimates. MultiCriticAL is tested in the context of multi-style learning, a special case of MTRL where agents are trained to behave with different distinct behavior styles, and yields up to 56% performance gains over the single-critic baselines and even successfully learns behavior styles in cases where single-critic approaches may simply fail to learn. In a simulated real-world use case, MultiCriticAL enables learning policies that smoothly transition between multiple fighting styles on an experimental build of EA's UFC game.

One-sentence Summary: MultiCriticAL is a single-actor, multi-critic framework for multi-task reinforcement learning, where task-based critic separation provides explicit per-task value-function approximation and enables improved performance over single-critic frameworks.

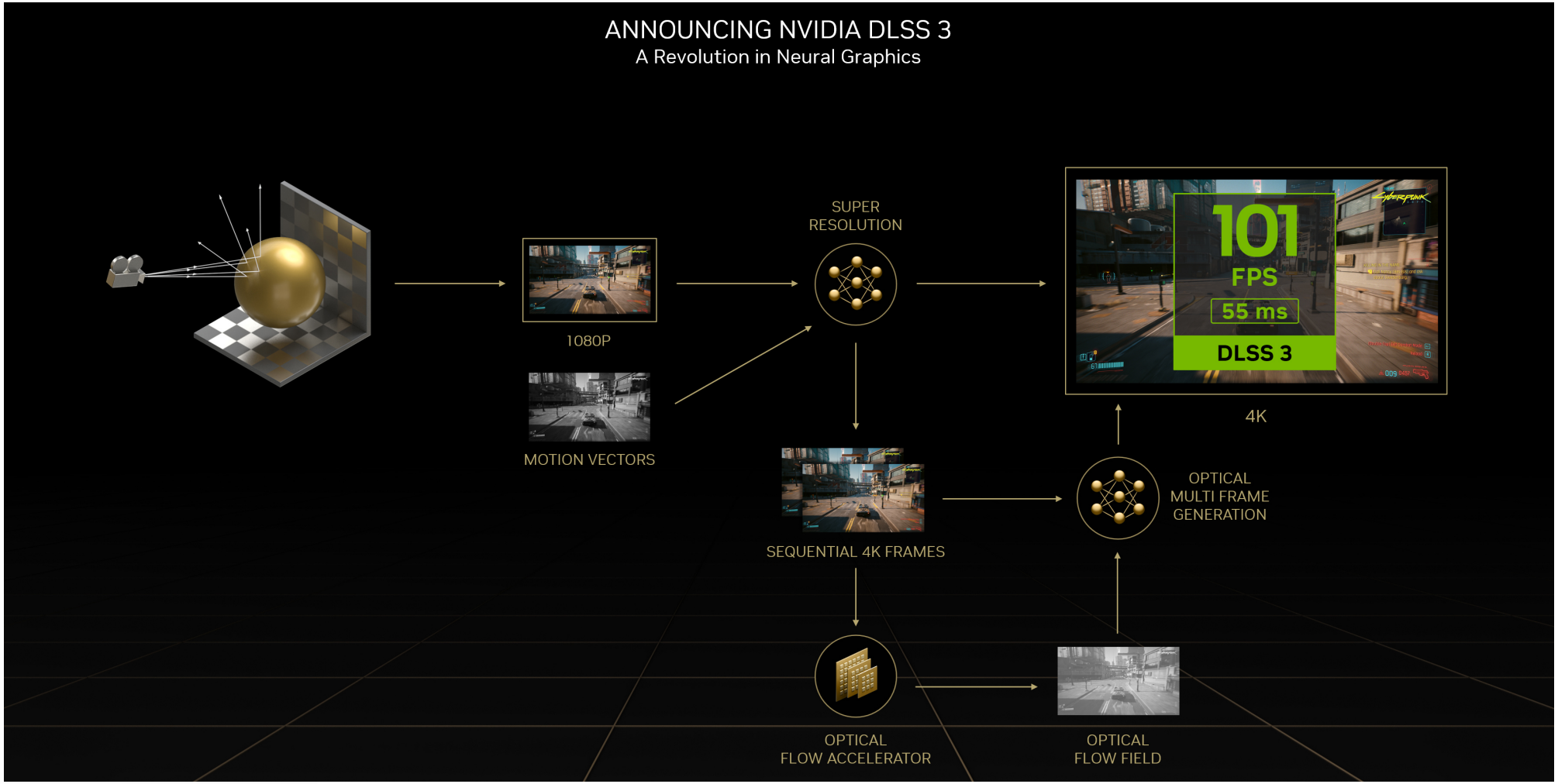
Supplementary Material: zip

Forza Series: Drivatar

https://web.archive.org/web/20070314015129/https://forzamotorsport.net/news/pitpassreports/pitpass36-2.htm

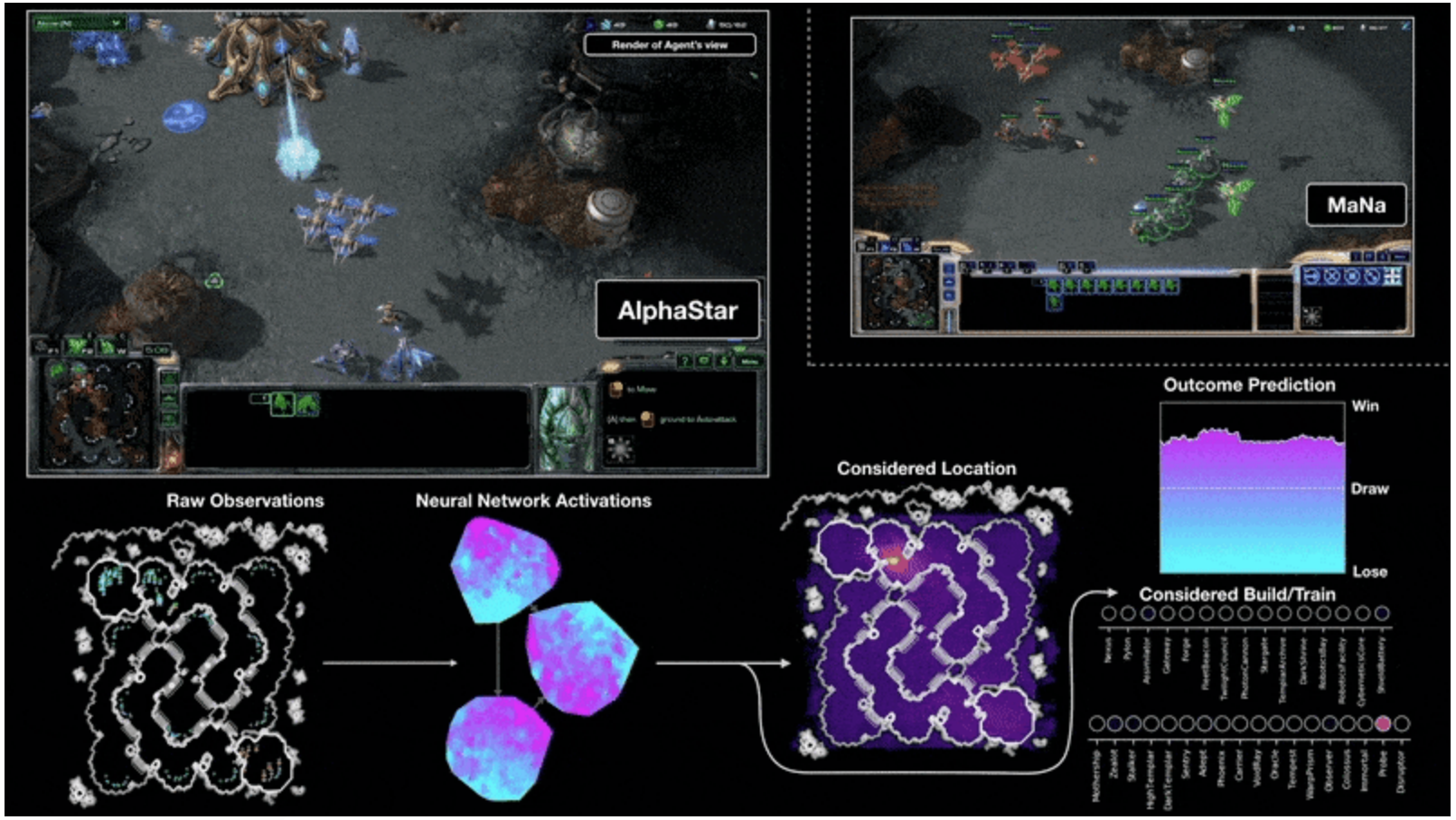


Nvidia DLSS 2/3



https://www.nvidia.com/es-la/geforce/news/dlss3-ai-powered-neural-graphics-innovations/

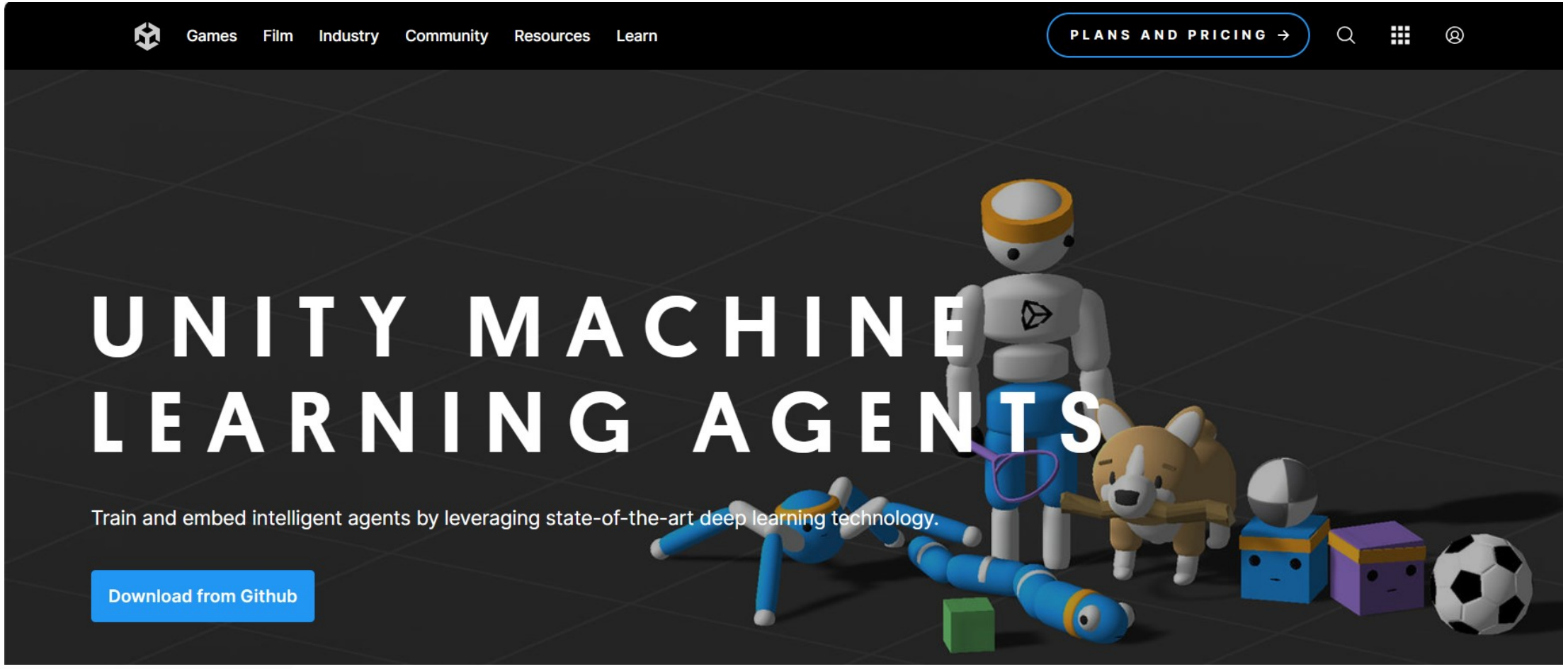
AlphaStar



https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii

Unity Machine Learning Agents

https://unity.com/products/machine-learning-agents



Research:

https://ieeexplore.ieee.org/abstract/document/8116624

Trained Behavior Trees: Programming by Demonstration to Support AI Game Designers

Ismael Sagredo-Olivenza, Pedro Pablo Gómez-Martín, Marco Antonio Gómez-Martín, and Pedro Antonio González-Calero

***Abstract*—Programming by demonstration (PbD) has a straightforward application in the development of the artificial intelligence (AI) for nonplayer characters (NPCs) in a video game: a game designer controls the NPC during a training session in the game, and thus demonstrates the expected behavior for that character in different situations. Afterwards, applying some machine learning technique on the traces recorded during the demonstration, an AI for the NPC can be generated. Nevertheless, with this approach, it is very hard for the game designer to fully control the resulting behavior, which is a key requirement for game designers, who are responsible for putting together a fun experience for the player. In this paper, we present trained behavior trees (TBTs). TBTs are behavior trees (BTs) generated from traces obtained in a game through PbD. BTs are a technique widely used for AI game programming that are created and modified through special purpose visual editors. By inducing a BT from a PbD game session, we com-**

such as "high general alertness (very nervous, thoroughly investigates every little disturbance, hard to circumvent, hard to shake off once he sees the target)."[1] Following the specifications given by designers, game AI programmers develop perception systems, behavior representation, path finding and decision making algorithms that conform with the AI system that game designers use to put together levels in the game.

AI game programmers and designers must work in close collaboration. Depending on the particular workflow of a given studio, designers may be in charge of just designing, tweaking parameters and testing the NPC behavior, or being responsible for programming part of the behavior through scripting, visual languages, or special purpose tools. Some solutions to AI prob-

https://ojs.aaai.org/index.php/AIIDE/article/view/18885

Proceedings of the Seventeenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment
(AIIDE 2021)

Reinforcement Learning Methods to Evaluate the Impact of AI Changes in Game Design

Pablo Gutiérrez-Sánchez,[1] Marco A. Gómez-Martín,[2]
Pedro A. González-Calero,[2] Pedro P. Gómez-Martín[2]

[1]PadaOne Games, Calle Profesor Jose Garcia Santesmases, 9, 28040, Madrid, Spain
[2]Complutense University of Madrid, Madrid, Spain,
pablo.gutierrez@padaonegames.com, marcoa@fdi.ucm.es, pagoncal@ucm.es, pedrop@fdi.ucm.es

Abstract

Game development has become a long process that requires many professionals working on a project during several months or years. With this scenario the re-utilization of resources is crucial not only to alleviate the process but also to bring coherence into the final product. In this paper we focus on the reuse of NPCs and the problems it brings about. In particular it is common to have different breeds (or personalities) of NPCs that are placed on different levels on the game. The problem arises when their behaviors are fine-tuned to accommodate a specific level needs without taking into consideration that this change may alter their performance on previous already-tested levels. The paper presents the application of reinforcement learning together with behavior trees to automatically test if modifications to the AIs of a stealth game have an impact on the user experience. Our experiments re-

efforts of human testers towards less mechanical and more creative tasks.

In this paper we focus on automatic regression tests for detecting issues when modifications are made to AIs that are reused in multiple sections of the game. These changes may be done when fine-tuning the behavior of a NPC in a level without thinking about the implications of those small variations on other levels where the NPC was placed before.

Changes as simple as slightly modifying an enemy's movement speed could end up unexpectedly impacting how the player interacts with the game's levels. These changes need not be particularly drastic (such as as sudden violations of the completability of a level), and may boil down to the player taking more or less time to complete a part of the game, or exploiting a new way to beat a level that was