Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Institute for AI and Autonomous Systems

Data-Driven Robot Software

How Learning-Based Components Fit Into the Puzzle

Dr. Alex Mitrevski
Master of Autonomous Systems

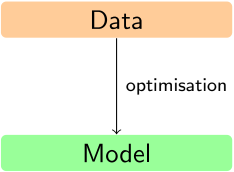
Structure

- Software development and machine learning
- Methods for data-driven development

Software Development and Machine Learning

What Do We Mean by Learning?

- Learning is a paradigm based on which **models — their parameters or structure — are extracted based on data**
- In robotics, we want a robot to **learn how to act by observing — and interacting with — the environment**
- Typically, learning is performed by **solving an optimisation problem over the model's parameters**



Why Learning in Robotics?

Reduced modelling effort

Learning-based approaches typically simplify the modelling effort when developing robot components and behaviours

Why Learning in Robotics?

Reduced modelling effort

Learning-based approaches typically simplify the modelling effort when developing robot components and behaviours

Increased behaviour flexibility

Learning-based behaviours can be more flexible than manually defined, particularly when trained on diverse data

Why Learning in Robotics?

Reduced modelling effort

Learning-based approaches typically simplify the modelling effort when developing robot components and behaviours

Increased behaviour flexibility

Learning-based behaviours can be more flexible than manually defined, particularly when trained on diverse data

Transferability

It can be easier to transfer learning-based components between robots — often achieved by retraining a model with robot-specific data, without implementation changes

Why Learning in Robotics?

Reduced modelling effort

Learning-based approaches typically simplify the modelling effort when developing robot components and behaviours

Increased behaviour flexibility

Learning-based behaviours can be more flexible than manually defined, particularly when trained on diverse data

Transferability

It can be easier to transfer learning-based components between robots — often achieved by retraining a model with robot-specific data, without implementation changes

Increased autonomy

In principle, learning-based robot components can be improved autonomously, without a need to make modifications to the underlying implementation

Simple Learning Example: Linear Regression

- Suppose that we are given **a set of n points** $\mathcal{D} = \{(x_i, y_i) \mid 1 \leq i \leq n\}$, and we want to **find the parameters of the best line that fits them**

Simple Learning Example: Linear Regression

- Suppose that we are given **a set of n points $\mathcal{D} = \{(x_i, y_i) \mid 1 \leq i \leq n\}$**, and we want to **find the parameters of the best line that fits them**

- We can represent a line by parameters $\boldsymbol{\theta} = (m, c)$ as $y = mx + c$, where m is a slope and c the intercept, and we can define the **mean-squared error as a loss function**:

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (mx_i + c))^2$$

Simple Learning Example: Linear Regression

- Suppose that we are given **a set of n points $\mathcal{D} = \{(x_i, y_i) \mid 1 \leq i \leq n\}$**, and we want to **find the parameters of the best line that fits them**

- We can represent a line by parameters $\boldsymbol{\theta} = (m, c)$ as $y = mx + c$, where m is a slope and c the intercept, and we can define the **mean-squared error as a loss function**:

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (mx_i + c))^2$$

- The best line fit can be found by solving the optimisation problem

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})$$

Simple Learning Example: Linear Regression

- Suppose that we are given **a set of n points $\mathcal{D} = \{(x_i, y_i) \mid 1 \leq i \leq n\}$**, and we want to **find the parameters of the best line that fits them**

- We can represent a line by parameters $\boldsymbol{\theta} = (m, c)$ as $y = mx + c$, where m is a slope and c the intercept, and we can define the **mean-squared error as a loss function**:

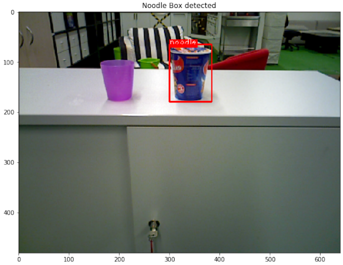
$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (mx_i + c))^2$$

- The best line fit can be found by solving the optimisation problem

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})$$

- This component can then be integrated into robot software, e.g. for wall detection from laser scan data during navigation

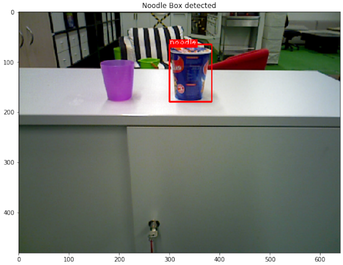
Another Learning Example: Object Classification



https://github.com/b-it-bots/mas_tutorials/blob/master/domestic_robotics/perception.ipynb

- Suppose now that we want to **classify objects in RGB images**; for this, we have a dataset $\mathcal{D} = \{(X_i, \boldsymbol{y}_i) \mid 1 \leq i \leq n\}$, where X_i is an image and $\boldsymbol{y}_i$ an object class label, represented as a one-hot encoding vector (assuming c classes)

Another Learning Example: Object Classification

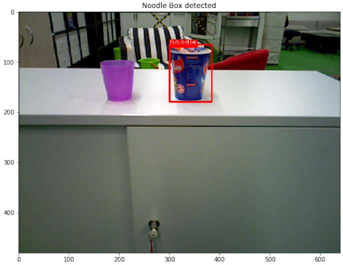


https://github.com/b-it-bots/mas_tutorials/blob/master/domestic_robotics/perception.ipynb

- Suppose now that we want to **classify objects in RGB images**; for this, we have a dataset $\mathcal{D} = \{(X_i, \boldsymbol{y}_i) \mid 1 \leq i \leq n\}$, where X_i is an image and $\boldsymbol{y}_i$ an object class label, represented as a one-hot encoding vector (assuming c classes)
- The classification can be performed by a neural network $y = f(X)$ with parameters $\boldsymbol{\theta}$, such that we can define the **cross-entropy loss function**:

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} \boldsymbol{y}_{i_j} \log(f(X)_j)$$

Another Learning Example: Object Classification



https://github.com/b-it-bots/mas_tutorials/blob/master/domestic_robotics/perception.ipynb

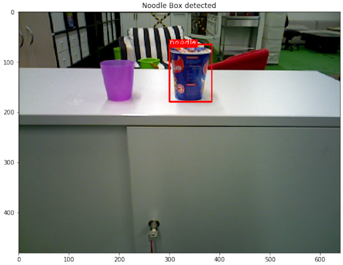
- Suppose now that we want to **classify objects in RGB images**; for this, we have a dataset $\mathcal{D} = \{(X_i, \boldsymbol{y}_i) \mid 1 \leq i \leq n\}$, where X_i is an image and $\boldsymbol{y}_i$ an object class label, represented as a one-hot encoding vector (assuming c classes)
- The classification can be performed by a neural network $y = f(X)$ with parameters $\boldsymbol{\theta}$, such that we can define the **cross-entropy loss function**:

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} \boldsymbol{y}_{i_j} \log(f(X)_j)$$

- As before, we can find appropriate parameters $\boldsymbol{\theta}$ by solving the optimisation problem

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})$$

Another Learning Example: Object Classification



https://github.com/b-it-bots/mas_tutorials/blob/master/domestic_robotics/perception.ipynb

- Suppose now that we want to **classify objects in RGB images**; for this, we have a dataset $\mathcal{D} = \{(X_i, \boldsymbol{y}_i) \mid 1 \leq i \leq n\}$, where X_i is an image and $\boldsymbol{y}_i$ an object class label, represented as a one-hot encoding vector (assuming c classes)
- The classification can be performed by a neural network $y = f(X)$ with parameters $\boldsymbol{\theta}$, such that we can define the **cross-entropy loss function**:

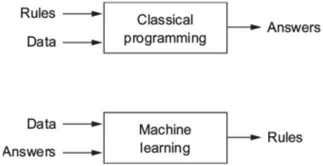
$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} \boldsymbol{y}_{i_j} \log(f(X)_j)$$

- As before, we can find appropriate parameters $\boldsymbol{\theta}$ by solving the optimisation problem

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})$$

- The classifier can then be integrated into a robot's scene understanding component

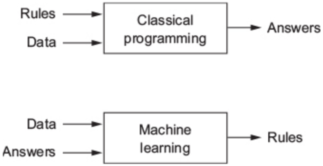
How Does This Relate to Robot Software Development?



F. Chollet and J. J. Allaire, “Deep Learning with R,” *Manning*, Jan. 2018, ch. 1.

- Complex robot behaviours are often difficult to develop — **learning can simplify this problem**
  - One of the main reasons why learning is so readily accepted by many roboticists

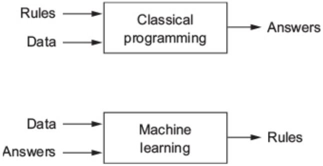
How Does This Relate to Robot Software Development?



F. Chollet and J. J. Allaire, "Deep Learning with R," *Manning*, Jan. 2018, ch. 1.

- Complex robot behaviours are often difficult to develop — **learning can simplify this problem**
  - One of the main reasons why learning is so readily accepted by many roboticists
- On a smaller scale, **it affects the development and integration of individual robot components**

How Does This Relate to Robot Software Development?



F. Chollet and J. J. Allaire, "Deep Learning with R," *Manning*, Jan. 2018, ch. 1.

- Complex robot behaviours are often difficult to develop — **learning can simplify this problem**
  - One of the main reasons why learning is so readily accepted by many roboticists
- On a smaller scale, **it affects the development and integration of individual robot components**
- In principle, learning can also **affect the complete development process**
  - Learning can be seen as a form of programming — the objective is not writing code, but the design of behaviours and the collection of data for learning those

Complex Robot Behaviours by Objective Maximisation

- Programming robots can replace explicit behaviour programming with a process of **learning to satisfy a mathematical objective function**, which involves:
  - **Defining an objective function** to be satisfied
  - **Training the robot** (in simulation or the real world) to optimise the objective function

Complex Robot Behaviours by Objective Maximisation

- Programming robots can replace explicit behaviour programming with a process of **learning to satisfy a mathematical objective function**, which involves:
  - **Defining an objective function** to be satisfied
  - **Training the robot** (in simulation or the real world) to optimise the objective function
- This reduces the software development burden, **letting the robot do the hard work** instead
  - Development effort is still needed for the optimisation algorithms

Complex Robot Behaviours Learned From Demonstration

- In many cases, it is easier to **demonstrate how to perform a task** than to write a program or to find a good objective function for it

Complex Robot Behaviours Learned From Demonstration

- In many cases, it is easier to **demonstrate how to perform a task** than to write a program or to find a good objective function for it
- Learning from demonstration requires two aspects:
  1. **Demonstrations by a human expert**
  2. A representation that **converts demonstrations into a form that a robot can use for subsequent execution**

Complex Robot Behaviours Learned From Demonstration

- In many cases, it is easier to **demonstrate how to perform a task** than to write a program or to find a good objective function for it
- Learning from demonstration requires two aspects:
  1. **Demonstrations by a human expert**
  2. A representation that **converts demonstrations into a form that a robot can use for subsequent execution**
- The primary benefit of learning from demonstrations is that **end users can influence the behaviour of the robot** (i.e. they can program the robot — without writing code)

Traditional vs. Data-Driven Software Development

Traditional

1. Design an **analytical model** of a desired behaviour
2. Decide on a **representation** for the behaviour (e.g. state machine)
3. Design a **software architecture** for implementing the behaviour
4. Create an **implementation** of the architecture

Traditional vs. Data-Driven Software Development

Traditional

1. Design an **analytical model** of a desired behaviour
2. Decide on a **representation** for the behaviour (e.g. state machine)
3. Design a **software architecture** for implementing the behaviour
4. Create an **implementation** of the architecture

Data-driven

1. Design an **optimisation criterion** for the desired behaviour
2. Decide on a **model class** for representing the behaviour (e.g. neural network)
3. Decide on **prior knowledge** to embed into the model to be learned
4. Create a **data collection strategy** for learning the behaviour
5. Collect data and learn the associated model

Traditional vs. Data-Driven Software Development

Traditional

1. Design an **analytical model** of a desired behaviour
2. Decide on a **representation** for the behaviour (e.g. state machine)
3. Design a **software architecture** for implementing the behaviour
4. Create an **implementation** of the architecture

Data-driven

1. Design an **optimisation criterion** for the desired behaviour
2. Decide on a **model class** for representing the behaviour (e.g. neural network)
3. Decide on **prior knowledge** to embed into the model to be learned
4. Create a **data collection strategy** for learning the behaviour
5. Collect data and learn the associated model

In both cases, the processes may need to be iterated, particularly to incorporate evaluation results

Integrating Learning Components Into Robot Programs

Multiple ways in which learning can be used in robotics:

Integrating Learning Components Into Robot Programs

Multiple ways in which learning can be used in robotics:

Learn parameters of existing analytical components

The simplest way to integrate learning — existing components do not need to be replaced, but the parameter selection is automated

Integrating Learning Components Into Robot Programs

Multiple ways in which learning can be used in robotics:

Learn parameters of existing analytical components

The simplest way to integrate learning — existing components do not need to be replaced, but the parameter selection is automated

Replace individual manually developed components with learning-based components

Concrete components of a robot's software (e.g. an object detector) are replaced by a learning-based counterpart — the overall architecture does not change much

Integrating Learning Components Into Robot Programs

Multiple ways in which learning can be used in robotics:

Learn parameters of existing analytical components

The simplest way to integrate learning — existing components do not need to be replaced, but the parameter selection is automated

Replace individual manually developed components with learning-based components

Concrete components of a robot's software (e.g. an object detector) are replaced by a learning-based counterpart — the overall architecture does not change much

Replace a complete pipeline with an end-to-end learning-based framework

The most intrusive approach in which a learning-based model (e.g. a single neural network or a sequence of networks) replaces a collection of developed components (e.g. a navigation pipeline)

Integrating Learning Components Into Robot Programs

Multiple ways in which learning can be used in robotics:

Learn parameters of existing analytical components

The simplest way to integrate learning — existing components do not need to be replaced, but the parameter selection is automated

Replace individual manually developed components with learning-based components

Concrete components of a robot's software (e.g. an object detector) are replaced by a learning-based counterpart — the overall architecture does not change much

Replace a complete pipeline with an end-to-end learning-based framework

The most intrusive approach in which a learning-based model (e.g. a single neural network or a sequence of networks) replaces a collection of developed components (e.g. a navigation pipeline)

Which of these is appropriate to use depends on multiple factors, such as **the maturity of analytical models**, **the data collection requirements**, and **the safety requirements**

How to Integrate Learning-Based Components Into Robot Software?

- The integration of learning components is typically easy — provided that the **software architecture is modular enough**
  - As long as the desired results of a behaviour are achieved, it should not matter how the behaviour is implemented

How to Integrate Learning-Based Components Into Robot Software?

- The integration of learning components is typically easy — provided that the **software architecture is modular enough**
  - As long as the desired results of a behaviour are achieved, it should not matter how the behaviour is implemented
- **Components to perform the actual learning process** need to be added
  - But learning is often performed offline, so the learning implementation is usually decoupled from the inference-time use of a learning-based component

How to Integrate Learning-Based Components Into Robot Software?

- The integration of learning components is typically easy — provided that the **software architecture is modular enough**
  - As long as the desired results of a behaviour are achieved, it should not matter how the behaviour is implemented
- **Components to perform the actual learning process** need to be added
  - But learning is often performed offline, so the learning implementation is usually decoupled from the inference-time use of a learning-based component
- Integration simplified by **de facto standard learning frameworks**
  - This encourages reuse-oriented development

Learning Component Integration Example[1]

Detection interface

```
class DetectionHandler(object):
    def __init__(self, detection_class,
                 class_annotation_file,
                 kwargs_file):
        self._detector = detection_class(class_file=class_annotation_file, model_kwargs_file=kwargs_file)

    def detect_objects(self, img):
        predictions = self._detector.detect([img])
        if len(predictions) < 1:
            raise RuntimeError('No predictions for image')
        bounding_boxes, classes, confidences \
            = ImageDetectorBase.prediction_to_bounding_boxes(predictions[0], self._detector.class_colors)
        return bounding_boxes, classes, confidences
```

High-level detection interface over ROS

```
class DetectionServer(object):
    def __init__(self, **kwargs):
        self._detection_handler = DetectionHandler(kwargs.get('detection_class', None),
                                                   kwargs.get('class_annotation_file', None),
                                                   kwargs.get('kwargs_file', None))

    def execute(self):
        cloud_msg = rospy.wait_for_message(self._cloud_topic, PointCloud2, timeout=self._timeout_s)
        img_msg = cloud_msg_to_image_msg(cloud_msg)
        bounding_boxes, classes, confidences = self._detection_handler.process_image_msg(img_msg)
```

[1] The examples on this slide are adapted from https://github.com/b-it-bots/mas_perception_libs

Is Traditional Software Development (Still) Useful in Robotics?

- It obviously is — that is why we are studying these things in the course
  - But we need to be open and keep an eye on data-driven frameworks that can ease our lives as robot software developers

Is Traditional Software Development (Still) Useful in Robotics?

- It obviously is — that is why we are studying these things in the course
  - But we need to be open and keep an eye on data-driven frameworks that can ease our lives as robot software developers
- Robot software should always be developed in a **modular fashion**
  - This simplifies any necessary integration of learning-based components

Is Traditional Software Development (Still) Useful in Robotics?

- It obviously is — that is why we are studying these things in the course
  - But we need to be open and keep an eye on data-driven frameworks that can ease our lives as robot software developers
- Robot software should always be developed in a **modular fashion**
  - This simplifies any necessary integration of learning-based components
- Traditional software development practices are also important in machine learning itself
  - Most open-source learning frameworks follow object-oriented programming principles

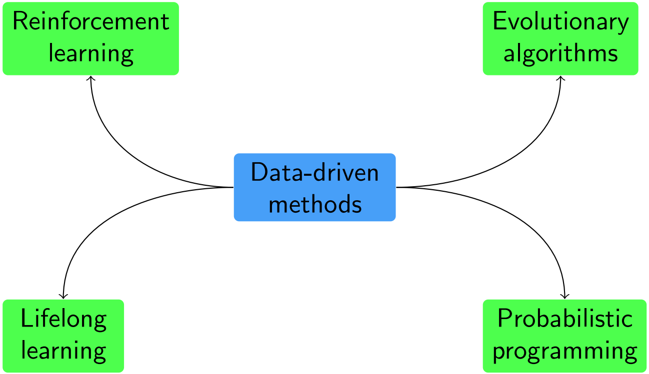
Is Traditional Software Development (Still) Useful in Robotics?

- It obviously is — that is why we are studying these things in the course
  - But we need to be open and keep an eye on data-driven frameworks that can ease our lives as robot software developers
- Robot software should always be developed in a **modular fashion**
  - This simplifies any necessary integration of learning-based components
- Traditional software development practices are also important in machine learning itself
  - Most open-source learning frameworks follow object-oriented programming principles
- **Using learning is not an invitation for developing bad software**
  - The interaction with learning-based components is still done through programs that we need to develop and maintain
  - Learning-based components need to be tested, deployed, and maintained just as any other software

Methods for Data-Driven Development

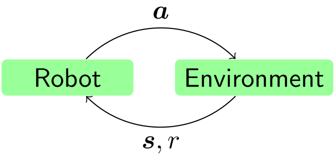
Methods Overview

- Various data-driven development methods can useful for robotics; we will look at the following ones on the next slides:



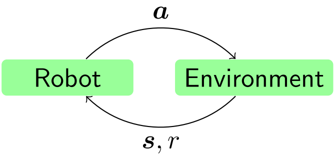
- Note that this figure is not a complete overview — more detailed information about learning frameworks is provided in the "Robot Learning" course

Reinforcement Learning (RL)



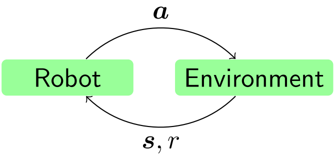
- The objective of RL is to learn an **execution policy** π **given observations of the form** $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}, r_t)$, where $\boldsymbol{s} \in S$ are states, $\boldsymbol{a} \in A$ are actions, and $r \in \mathbb{R}$ is a reward

Reinforcement Learning (RL)



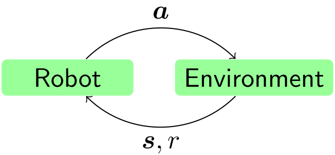
- The objective of RL is to learn an **execution policy** π **given observations of the form** $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}, r_t)$, where $\boldsymbol{s} \in S$ are states, $\boldsymbol{a} \in A$ are actions, and $r \in \mathbb{R}$ is a reward
- The robot's behaviour is then **fully controlled by the learned policy**

Reinforcement Learning (RL)



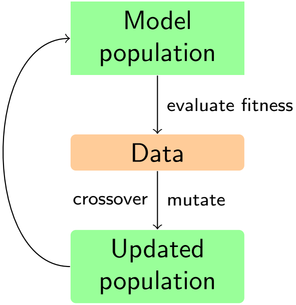
- The objective of RL is to learn an **execution policy π given observations of the form $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}, r_t)$**, where $\boldsymbol{s} \in S$ are states, $\boldsymbol{a} \in A$ are actions, and $r \in \mathbb{R}$ is a reward
- The robot's behaviour is then **fully controlled by the learned policy**
- Reinforcement learning can be observed as automatic program synthesis, as **the policy is acquired from data rather than being manually defined**

Reinforcement Learning (RL)



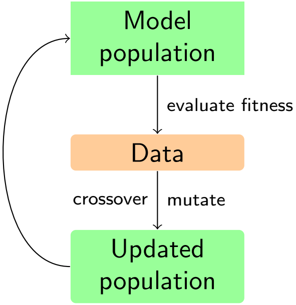
- The objective of RL is to learn an **execution policy π given observations of the form $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}, r_t)$**, where $\boldsymbol{s} \in S$ are states, $\boldsymbol{a} \in A$ are actions, and $r \in \mathbb{R}$ is a reward
- The robot's behaviour is then **fully controlled by the learned policy**
- Reinforcement learning can be observed as automatic program synthesis, as **the policy is acquired from data rather than being manually defined**
- RL is widely used in robotics

Evolutionary Algorithms



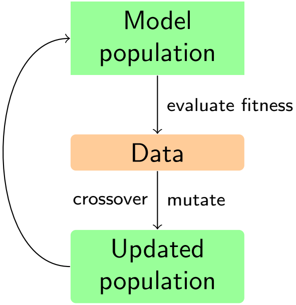
- Evolutionary optimisation is a somewhat related framework in which **a set of individuals** (candidate models) **are preserved during learning**

Evolutionary Algorithms



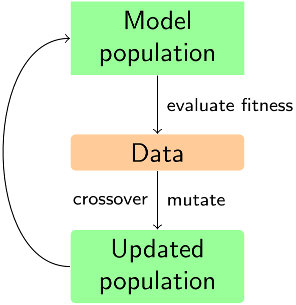
- Evolutionary optimisation is a somewhat related framework in which **a set of individuals** (candidate models) **are preserved during learning**
- The **fitness of the individuals** is evaluated based on data — or any other process knowledge — and **the population is updated** from the most fit individuals

Evolutionary Algorithms



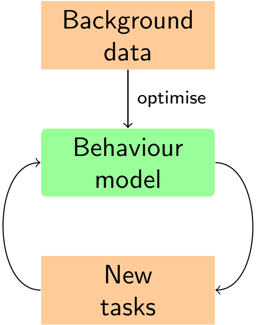
- Evolutionary optimisation is a somewhat related framework in which **a set of individuals** (candidate models) **are preserved during learning**
- The **fitness of the individuals** is evaluated based on data — or any other process knowledge — and **the population is updated** from the most fit individuals
- This process can result in either **a single model to drive the behaviour** or **a collection of behaviour models** — potentially suitable for different cases

Evolutionary Algorithms



- Evolutionary optimisation is a somewhat related framework in which **a set of individuals** (candidate models) **are preserved during learning**
- The **fitness of the individuals** is evaluated based on data — or any other process knowledge — and **the population is updated** from the most fit individuals
- This process can result in either **a single model to drive the behaviour** or **a collection of behaviour models** — potentially suitable for different cases
- Evolutionary optimisation is also a form of automatic program synthesis and can be used for **optimising model structure**

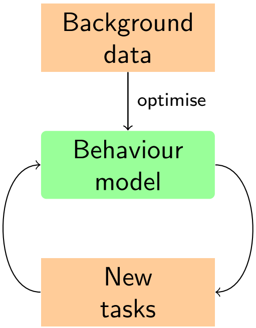
Neural Networks (NNs) and Lifelong Learning



- When representing robot behaviour, NNs can be seen as programs that **process an input, such as sensor data, to produce an output, such as robot motions**
  - In fact, the RL policy is often represented by an NN

[2] R. Firoozi et al., "Foundation Models in Robotics: Applications, Challenges, and the Future", *CoRR*, vol. abs/2312.07843, Dec. 2023. Available: https://arxiv.org/abs/2312.07843

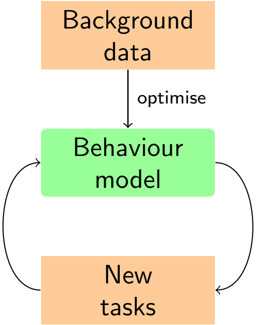
Neural Networks (NNs) and Lifelong Learning



- When representing robot behaviour, NNs can be seen as programs that **process an input, such as sensor data, to produce an output, such as robot motions**
  - In fact, the RL policy is often represented by an NN
- An NN is typically pretrained and then specialised on task-specific data
  - If the model is pretrained on a very large dataset, it is referred to as a **foundation model**[2]

[2] R. Firoozi et al., "Foundation Models in Robotics: Applications, Challenges, and the Future", *CoRR*, vol. abs/2312.07843, Dec. 2023. Available: https://arxiv.org/abs/2312.07843

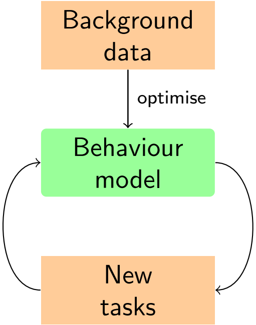
Neural Networks (NNs) and Lifelong Learning



- When representing robot behaviour, NNs can be seen as programs that **process an input, such as sensor data, to produce an output, such as robot motions**
  - In fact, the RL policy is often represented by an NN
- An NN is typically pretrained and then specialised on task-specific data
  - If the model is pretrained on a very large dataset, it is referred to as a **foundation model**[2]
- Different ways to incorporate data about new tasks into a model:

[2] R. Firoozi et al., "Foundation Models in Robotics: Applications, Challenges, and the Future", *CoRR*, vol. abs/2312.07843, Dec. 2023. Available: https://arxiv.org/abs/2312.07843

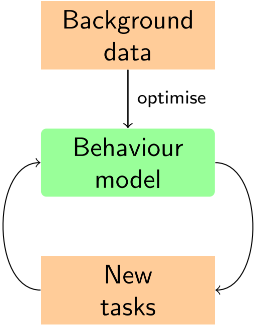
Neural Networks (NNs) and Lifelong Learning



- When representing robot behaviour, NNs can be seen as programs that **process an input, such as sensor data, to produce an output, such as robot motions**
  - In fact, the RL policy is often represented by an NN
- An NN is typically pretrained and then specialised on task-specific data
  - If the model is pretrained on a very large dataset, it is referred to as a **foundation model**[2]
- Different ways to incorporate data about new tasks into a model:
  - **Finetuning the model on new data**: The model is not used for the original tasks anymore

[2] R. Firoozi et al., "Foundation Models in Robotics: Applications, Challenges, and the Future", *CoRR*, vol. abs/2312.07843, Dec. 2023. Available: https://arxiv.org/abs/2312.07843

Neural Networks (NNs) and Lifelong Learning



- When representing robot behaviour, NNs can be seen as programs that **process an input, such as sensor data, to produce an output, such as robot motions**
  - In fact, the RL policy is often represented by an NN
- An NN is typically pretrained and then specialised on task-specific data
  - If the model is pretrained on a very large dataset, it is referred to as a **foundation model**[2]
- Different ways to incorporate data about new tasks into a model:
  - **Finetuning the model on new data**: The model is not used for the original tasks anymore
  - **Incorporating data about new tasks into the model**: Information about old tasks is explicitly preserved — the model is expected to perform well both on old and new tasks

[2] R. Firoozi et al., "Foundation Models in Robotics: Applications, Challenges, and the Future", *CoRR*, vol. abs/2312.07843, Dec. 2023. Available: https://arxiv.org/abs/2312.07843

Probabilistic Programming

- Probabilistic programming is **a programming paradigm in which programs are observed as probabilistic models**
  - The program thus models a (usually complex) probability distribution

[3] For instance Pyro (https://pyro.ai/), which is embedded in Python.

Probabilistic Programming

- Probabilistic programming is **a programming paradigm in which programs are observed as probabilistic models**
  - The program thus models a (usually complex) probability distribution

- The objective is to enable the specification of Bayesian probabilistic models that can be used for **generating samples** and **performing inference** based on prior knowledge and observed data

[3] For instance Pyro (https://pyro.ai/), which is embedded in Python.

Probabilistic Programming

- Probabilistic programming is **a programming paradigm in which programs are observed as probabilistic models**
  - The program thus models a (usually complex) probability distribution
- The objective is to enable the specification of Bayesian probabilistic models that can be used for **generating samples** and **performing inference** based on prior knowledge and observed data
- There exist **probabilistic programming languages** that simplify the development of probabilistic programs[3]

[3] For instance Pyro (https://pyro.ai/), which is embedded in Python.

Probabilistic Programming

- Probabilistic programming is **a programming paradigm in which programs are observed as probabilistic models**
  - The program thus models a (usually complex) probability distribution
- The objective is to enable the specification of Bayesian probabilistic models that can be used for **generating samples** and **performing inference** based on prior knowledge and observed data
- There exist **probabilistic programming languages** that simplify the development of probabilistic programs[3]
- **Probabilistic reasoning is at the core of intelligent robots** (see AMR), and probabilistic programming can simplify the integration of probabilistic models into robot programs

[3] For instance Pyro (https://pyro.ai/), which is embedded in Python.

Data Engineering


Original data


Augmented data

- An essential process of every data-driven framework
  - Without it, we risk having garbage-in, garbage out

Data Engineering

Original data

Augmented data

- An essential process of every data-driven framework
  - Without it, we risk having garbage-in, garbage out
- Various aspects of learning data need to be verified for learning to be successful:

Data Engineering

Original data

Augmented data

- An essential process of every data-driven framework
  - Without it, we risk having garbage-in, garbage out
- Various aspects of learning data need to be verified for learning to be successful:
  - **Quality**: Is the data representative of the learning problem and are there any problems with the collection procedure?

Data Engineering

Original data

Augmented data

- An essential process of every data-driven framework
  - Without it, we risk having garbage-in, garbage out
- Various aspects of learning data need to be verified for learning to be successful:
  - **Quality**: Is the data representative of the learning problem and are there any problems with the collection procedure?
  - **Coverage**: Is the data diverse enough for a general model to be learned?

Data Engineering

Original data

Augmented data

- An essential process of every data-driven framework
  - Without it, we risk having garbage-in, garbage out
- Various aspects of learning data need to be verified for learning to be successful:
  - **Quality**: Is the data representative of the learning problem and are there any problems with the collection procedure?
  - **Coverage**: Is the data diverse enough for a general model to be learned?
  - **Bias**: Does the data contain biases that may skew a learned model in an undesired way?

Data Engineering

Original data

Augmented data

- An essential process of every data-driven framework
  - Without it, we risk having garbage-in, garbage out
- Various aspects of learning data need to be verified for learning to be successful:
  - **Quality**: Is the data representative of the learning problem and are there any problems with the collection procedure?
  - **Coverage**: Is the data diverse enough for a general model to be learned?
  - **Bias**: Does the data contain biases that may skew a learned model in an undesired way?
- Short of a larger real dataset, **data augmentation** is typically used to increase the coverage of training data
  - Augmentation is a process in which existing data items are used to create new, artificial training examples

What Do We Lose With Data-Driven Development?

Full control over and understanding of our systems

- Learning-based components — particularly those based on neural networks — can exhibit **emergent behaviours** (behaviours that they are not directly trained for)
- Although it can be argued that very large (non-learning-based) software projects can also be too complex to understand (at least without significant effort)

What Do We Lose With Data-Driven Development?

Full control over and understanding of our systems

- Learning-based components — particularly those based on neural networks — can exhibit **emergent behaviours** (behaviours that they are not directly trained for)
- Although it can be argued that very large (non-learning-based) software projects can also be too complex to understand (at least without significant effort)

Simple problem resolution

- Barring implementation issues in the learning algorithms, problems in data-driven components are typically resolved by retraining with additional data
- Lifelong learning frameworks pose an additional challenge — retraining from scratch will eliminate whatever has been learned online

What Do We Lose With Data-Driven Development?

Full control over and understanding of our systems

- Learning-based components — particularly those based on neural networks — can exhibit **emergent behaviours** (behaviours that they are not directly trained for)
- Although it can be argued that very large (non-learning-based) software projects can also be too complex to understand (at least without significant effort)

Simple problem resolution

- Barring implementation issues in the learning algorithms, problems in data-driven components are typically resolved by retraining with additional data
- Lifelong learning frameworks pose an additional challenge — retraining from scratch will eliminate whatever has been learned online

The fun and creativity of software development

Developing large software can be a very creative task — relying on data-driven frameworks can simplify the development, but also reduce some of the task's creativity

Robot Software Development in Future (Own Opinion)

- Learning is **an essential piece of the puzzle for general-purpose, adaptive robots**
  - Cognition is limited without learning
  - Learning is likely to become a standard element of robot software development

Robot Software Development in Future (Own Opinion)

- Learning is **an essential piece of the puzzle for general-purpose, adaptive robots**
  - Cognition is limited without learning
  - Learning is likely to become a standard element of robot software development
- **The potentials of learning for robotics** have to be acknowledged
  - Learning makes it possible to implement robot functionalities that have otherwise been very difficult to achieve

Robot Software Development in Future (Own Opinion)

- Learning is **an essential piece of the puzzle for general-purpose, adaptive robots**
  - Cognition is limited without learning
  - Learning is likely to become a standard element of robot software development
- **The potentials of learning for robotics** have to be acknowledged
  - Learning makes it possible to implement robot functionalities that have otherwise been very difficult to achieve
- But it is important to **recognise the limitations of learning-based frameworks** with respect to safety-critical systems
  - We can only develop better learning-based systems by actively using learning in robotics and identifying problems that we can solve