Machine Learning as A New Tool for Applied Mathematicians

A Tutorial

Nan Ye

School of Mathematics and Physics
The University of Queensland

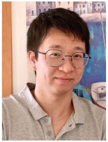
ANZIAM 2023, Cairns

- Nan Ye, School of Mathematics and Physics, UQ
- Research: Turn data into insights, predictions and decisions.
  - broad interest in AI/ML/Stat/DS
  - theory and algorithms: sequential decision making, statistical learning theory, numerical optimization, Bayesian learning
  - applications: autonomous driving, understanding routing behavior, cyber attack detection, fishery management.
- Group members

me
(with wife and daughter)

Marcus Hoerger
planning under uncertainty

Jun Ju
reinforcement learning



Yeming Lei
fishery stock assessment

Jonathan Wilton
weakly supervised learning

The Journey Begins

capabilities
a growing frontier

machine learning 102
maths and intuition

PDE solver
science is a differential equation
Alan Turing

discrete optimization
used by
50% Edelman Award winners
George Nemhauser

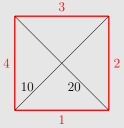
time series modelling
time is the only thing
Miles Davis

roadmap
a path forward

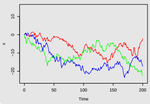
machine learning has many applications in maths

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2},$$
$$u(0,x) = -\sin(\pi, x),$$
$$u(t,-1) = u(t,1) = 0,$$

PDEs



combinatorial optimization



time series

$\alpha \to (\beta \to \alpha)$
$((\neg\alpha) \to (\neg\beta)) \to (\beta \to \neg\alpha)$
$\dots$
$\alpha, \alpha \to \beta \vdash \beta$

theorem proving

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 4 & 5 \end{pmatrix}$$

matrix multiplication

and many others...

world $\to$ model $\to$ solve $\to$ validate $\to$ deploy

mathematical modelling

where machine learning comes in

- new fundamental tools (e.g., faster matrix multiplication algorithms)
- new function representations (e.g., neural networks, random forests)
- new ways of solving existing models (e.g., neural PDE solvers)

capabilities
a growing frontier

machine learning 102
maths and intuition

PDE solver
science is a differential equation
Alan Turing

discrete optimization
used by
50% Edelman Award winners
George Nemhauser

time series modelling
time is the only thing
Miles Davis

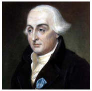
roadmap
a path forward

Theorem. Machine learning has many core ideas rooted in classical maths.

Proof by a fictitious story: *One day, many great mathematicians meet at Cairns, and a very curious turtle poses this question to them...*

what's the shape of the wave?

Weierstrass:
polynomials are good approximations



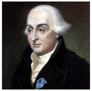
Lagrange:
perfect measurements $\Rightarrow$ use my interpolating plynomial

Gauss:
perfect or noisy $\Rightarrow$ use my method of least squares

Gerstner:
use my analytical solution to Euler's equation

what's the shape of the wave?

Weierstrass:
polynomials are good approximations



Lagrange:
perfect measurements $\Rightarrow$ use my interpolating plynomial

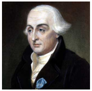
Gauss:
perfect or noisy $\Rightarrow$ use my method of least squares

machine learning approach

Gerstner:
use my analytical solution to Euler's equation

what's the shape of the wave?

Weierstrass:
polynomials are good approximations



Lagrange:
perfect measurements $\Rightarrow$ use my interpolating plynomial

Gauss:
perfect or noisy $\Rightarrow$ use my method of least squares

machine learning approach

Gerstner:
use my analytical solution to Euler's equation

classical maths approach

Statistical Learning

- We assume a fixed but unknown distribution $p(X, Y)$ on the input X and the output Y.
- Each model $f : \mathcal{X} \to \mathcal{Y}$ belongs to a model class $\mathcal{F}$.
- Objective: find $f \in \mathcal{F}$ minimizing the expected risk

  $$R(f) = \mathbb{E}_p \, \ell(f(X), Y),$$

  where the loss function $\ell : Y \times Y \to \mathbb{R}^{\geq 0}$ measures how well $f(X)$ agrees with Y.
- Expected risk cannot be computed $\Rightarrow$ estimate using empirical risk

  $$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).$$

“Mother” of learning algorithms

- Regularized empirical risk minimization:

$$\min_{f \in \mathcal{F}} \left[\hat{R}(f) + \lambda C(f) \right],$$

where

  - $C(f)$ is a complexity measure for f.
  - $\lambda \geq 0$ is the regularization constant.

- Intuitively, find f that fits data well and is simple.

learning

=

data + model class + fitness + complexity measure + optimization

Examples

regression

linear regression

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} (x_i^T \mathbf{w} - y_i)^2.$$

ridge regression

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} (x_i^T \mathbf{w} - y_i)^2 + \lambda ||\mathbf{w}||_2^2.$$

LASSO

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} (x_i^T \mathbf{w} - y_i)^2 + \lambda ||\mathbf{w}||_1^2.$$

classification

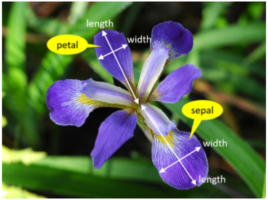
logistic regression

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} \ln(1 + e^{-y_i x_i^T \mathbf{w}}) + \lambda ||\mathbf{w}||_2^2.$$

SVM

$$\min_{\mathbf{w}\in\mathbf{R}^d, b\in\mathbf{R}} \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(x_i^T \mathbf{w} + b)).$$

Hands-on



sepal		petal		
length	width	length	width	class
5.1	3.5	1.4	0.2	setosa
7.	3.2	4.7	1.4	versicolor
6.3	3.3	6.	2.5	virginica
		...		

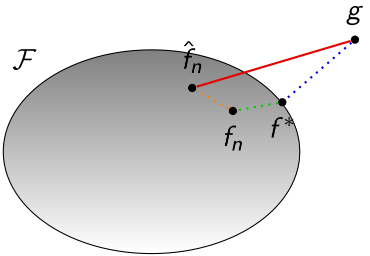
try me: https://tinyurl.com/272nbkmy

```python
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X, y = load_iris(return_X_y=True)
X_tr, X_ts, y_tr, y_ts= train_test_split(X, y, test_size=0.3,
    random_state=42)
reg = LogisticRegression().fit(X_tr, y_tr)
print("MSE (train) = ", reg.score(X_tr, y_tr))
print("MSE (test) = ", reg.score(X_ts, y_ts))
```

Making It Work Well

error $\leq$ approximation error + estimation error + optimization error



g: ground truth
f^*: optimal approximation in $\mathcal{F}$
f_n: optimal fit on data
$\hat{f}_n$: computed fit on data

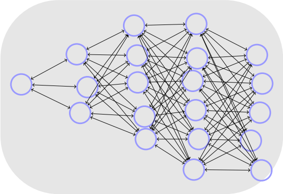
- Choose the model class carefully
  - expressivity of $\mathcal{F} \uparrow \Rightarrow$ approximation error $\downarrow$, estimation error $\uparrow$
  - choose a simple model class using domain knowledge if possible.
- Computing a sub-optimal fit may lead to better generalization.
  - particularly when $\mathcal{F}$ is very complex

model selection approaches

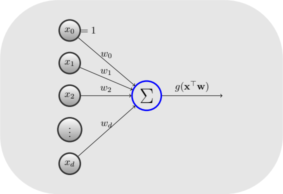
- use data to estimate each option's generalization performance
  - validation set, cross validation, bootstrapping
- analytically approximate the generalization performance
  - AIC, BIC, MDL

Neural Networks (NNs)

biological NN



artificial NN (ANN)



artificial neuron

- ANNs
  - interconnected simple computational units (neurons)
  - universal approximators
  - often trained to minimize loss
- Neurons
  - input from incoming edges, output along outgoing edges
  - computes nonlinearly transformed weighted input sum $g(\mathbf{w}^\top \mathbf{x})$
  - nonlinearity g known as activation/transfer function

architecture	**activation**	**optimizer**	**software**
MLP	threshold	SGD	PyTorch
CNN	sigmoid	AdaGrad	TensorFlow
RNN	ReLU	RMSprop	Google JAX
ResNet	ELU	AdaDelta	Keras
transformer	GELU	Adam	MXNet
...	...	...	...

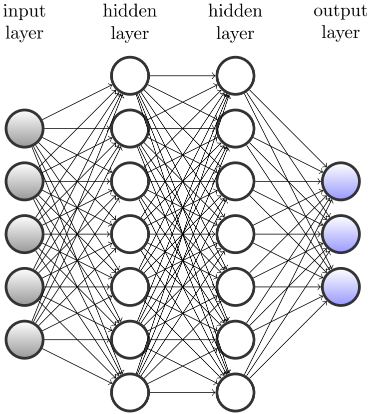
$\uparrow$

often first-order methods

gradients computed using automatic differentiation

Multilayer Perceptron (MLP)

aka multilayer feedforward neural network

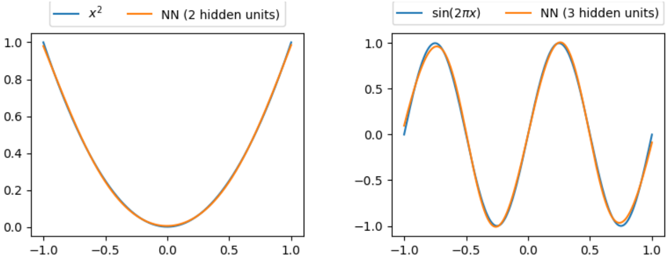


- neurons organized in layers
- forward edges only (from input neurons to output neurons)
- single-hidden layer sigmoid MLPs are universal approximators

Universal approximation property of single hidden neural net

$$\sum_{i=1}^{m} \alpha_i \sigma(w_i x + b_i) + \beta,$$

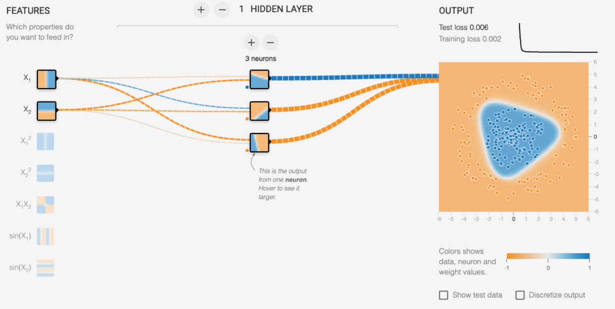
where $\sigma(u) = 1/(1 + e^{-x})$ *is the sigmoid function.*



$$x^2 \approx 2.2\sigma(-3.15x - 3) + 2.2\sigma(3.15x - 3) - 0.205, \; x \in [-1, 1].$$

$$\sin(x) \approx 10.9\sigma(-6.35x - 3.05) - 10.9\sigma(6.35x - 3.05) - 36.6\sigma(-1.3x) + 18.23, \; x \in [-1, 1].$$

Feature Learning



https://playground.tensorflow.org/

a sigmoid unit approximately learns the concept of a circular area in 2D plane

- In deep neural networks ($>$ 1 hidden layer), deeper layers are capable of learning higher-level features.
- This allows learning accurate models from raw features without handcrafting high-level features.

Hands-on

try me: https://tinyurl.com/27vvyrky

```
net = nn.Sequential(nn.Linear(2, 10), nn.ReLU(), nn.Linear(10, 1))
optimizer = optim.SGD(net.parameters(), lr=0.5, momentum=0)
mse = MSELoss()
for i in range(200):
  optimizer.zero_grad()
  loss = mse(net(X), Y)
  loss.backward()
  optimizer.step()
```

learn a single-hidden layer neural network $f_{\mathbf{w}}(\mathbf{x})$ by minimizing its mean squared error on the training set

capabilities
a growing frontier

machine learning 102
maths and intuition

PDE solver
science is a differential equation
Alan Turing

discrete optimization
used by
50% Edelman Award winners
George Nemhauser

time series modelling
time is the only thing
Miles Davis

roadmap
a path forward

PDEs in Classical Maths

domain	PDE	solution method
fluid flow	Navier-Stokes	analytic methods
	Poiseuille	finite difference
	Couette	finite volume
electromagnetism	Maxwell's	finite element
epidemiology	SIR	Runge-Kutta
...		...

Physics-Informed Machine Learning

Applications

- Cardiac simulation (Zhang et al., 2022)
- 4D-flow MRI (Kissas et al., 2020)
- Seismic wave (Karimpouli and Tahmasebi, 2020)
- NVIDIA Modulus (previously SimNet) for multi-physics simulation (Hennigh et al., 2021)
- Material sciences, molecular simulations, geophysics, ...

Inverse problem: data → model

what's the shape of the wave?

- Data-driven: fit a model using data only
- Physics-driven: use the governing equation to obtain a solution consistent with the data
- Physics-informed: fit a model consistent with both data and physics

$$\min_{f \in \mathcal{F}} [\hat{R}_{\text{data}}(f) + \mu \hat{R}_{\text{physics}}(f)]$$

  - Physics-based loss $\hat{R}_{\text{physics}}$ measures how much physical laws are violated at selected points.
  - Physics-Informed Neural Networks (PINN): model is an NN

Forward problem: model $\rightarrow$ data

- When boundary/initial conditions, instead of data, are given,

$$\min_{f \in \mathcal{F}} [\hat{R}_{\text{boundary}}(f) + \mu \hat{R}_{\text{physics}}(f)]$$

  - The loss $\hat{R}_{\text{boundary}}$ measures how much boundary/initial conditions are violated at selected points.

Example: an inverse problem

$$u_{tt} = c^2 u_{xx}, \qquad (x,t) \in [0, 2\pi] \times [0, 10],$$
$$\text{observations}: \quad \{(x_i, t_i, u_i)\}_{i=1}^{n}.$$

- Physics-informed machine learning:

$$\min_{f \in \mathcal{F}} \left[\frac{1}{n} \sum_{i=1}^{n} (u(x_i, t_i) - u_i)^2 + \mu \frac{1}{m} \sum_{i=1}^{n} r(x_i', t_i')^2 \right],$$

  - $r = u_{tt} - c^2 u_{xx}$ is the PDE residual,
  - $\{(x_i', t_i')\}_{i=1}^{m}$ are selected points in the domain.
- Gradient-based optimization methods can be applied to solve the optimization problem, with all derivatives computed using automatic differentiation.

Example: a forward problem / simulation

$$\begin{aligned} u_{tt} &= c^2 u_{xx}, & (x,t) &\in [0,2\pi]\times[0,10], \\ u(x,0) &= \sin(x), & x &\in [0,2\pi] \\ u_t(x,0) &= 0, & x &\in [0,2\pi]. \end{aligned}$$

- Physics-informed machine learning:

$$\min_{f\in\mathcal{F}} \left[\frac{1}{n}\sum_{i=1}^{n}(u(x_i,0)-\sin(u_i))^2 + \frac{1}{n}\sum_{i=1}^{n}(u_t(x_i,0))^2 + \mu\frac{1}{m}\sum_{i=1}^{n} r(x_i',t_i')^2 \right],$$

  - $\{(x_i\}_{i=1}^{n}$ and $\{(x_i',t_i')\}_{i=1}^{m}$ are selected points in the domain.

capabilities
a growing frontier

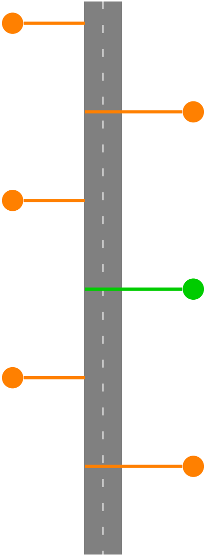
machine learning 102
maths and intuition

PDE solver
science is a differential equation
Alan Turing

discrete optimization
used by
50% Edelman Award winners
George Nemhauser

time series modelling
time is the only thing
Miles Davis

roadmap
a path forward



Classical Discrete Optimization

- Discrete optimization problem is everywhere: travelling salesman problem (TSP), vehicle routing, data center resource management, timetable scheduling, planning a trip to Cairns...
- Discrete optimization problems are often intractable in general.
- Classical solution methods often rely on problem-specific heuristics, discovered by experts over time.
- Classical solution software often needs to be properly configured to get the best results.

Automation with Machine Learning

Applications

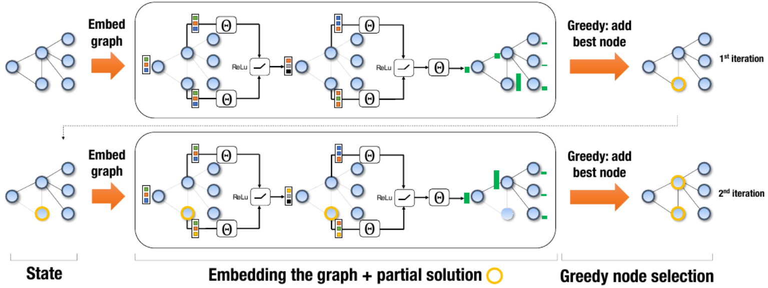
- Configure algorithms (e.g., configure CPLEX hyperparameters)
- Learn an end-to-end solution (e.g., planar TSP, planar convex hulls)
- Learn greedy heuristics for decisions in an algorithm (e.g., deciding which node to travel to in TSP)

Learning paradigms

- Imitation learning
  - build a training set of good algorithmic decisions, and learn using a supervised learning algorithm
  - example: learn to efficiently approximate expensive branching decisions in branch-and-bound (Alvarez, Louveaux, and Wehenkel, 2017; Gasse et al., 2019)
- Reinforcement learning (RL)
  - AlphaGo and ChatGPT use RL
  - specify the problem and when reward/penalty is given, do many trial and error, and gradually improve the solution strategy
  - example: learn new next node selection strategy in TSP (Dai et al., 2017)

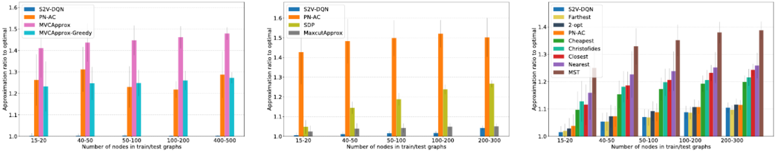
making it work: good features + good data (+ good reward for RL)

Example



(Dai et al., 2017)

- handcrafting features is hard ⇒ learn features using graphical neural networks
- use reinforcement learning to learn both the features and the node selection strategy

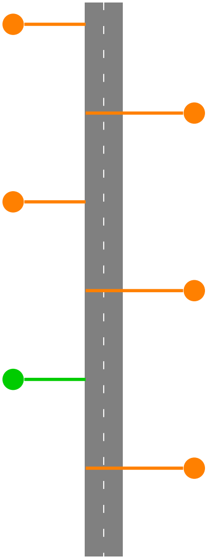


(a) MVC BA (b) MAXCUT BA (c) TSP random

(Dai et al., 2017)

- Problems: minimum vertex cover, maximum cut, TSP
- The machine learning approach (S2V-DQN) outperforms strong approximation algorithms.

capabilities
a growing frontier



machine learning 102
maths and intuition

PDE solver
science is a differential equation
Alan Turing

discrete optimization
used by
50% Edelman Award winners
George Nemhauser

time series modelling
time is the only thing
Miles Davis

roadmap
a path forward

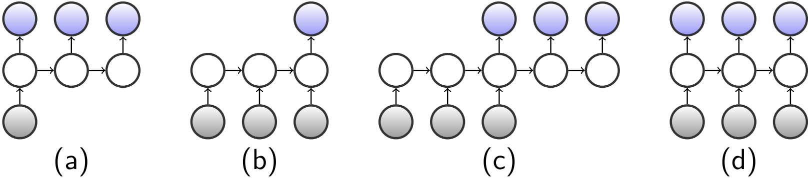
Classical Time Series Models

- Classical time series models like AR, ARMA, ARIMA model recurrent relationships between current and past.
- For example, in ARMA(p, q)

$$x_t = \phi_1 x_{t-1} + \ldots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \ldots + \theta_q w_{t-q}.$$

  $\Rightarrow$ this is moving linear regression.
- These are limited in their expressivity.

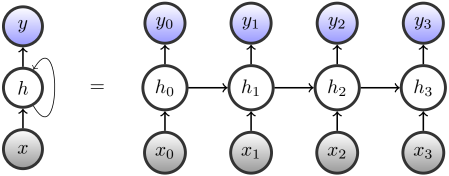
Recurrent Neural Networks (RNNs)



RNNs are good for various sequence modelling problems, including

(a) One to many, e.g. image captioning
(b) Many to one, e.g. video classification
(c) Many to many, e.g. machine translation
(d) Many to many, e.g. video frame classification

- The states of hidden neurons in an RNN are updated at each time step.
- For finite sequences, RNNs can be *unfolded* as feedforward networks

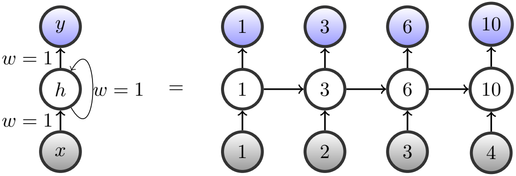


- The slices at all time steps share the same parameters W

$$h_t = f_W(h_{t-1}, x_t),$$
$$y_t = g_W(h_t).$$

Example: An RNN for summing a sequence

- The RNN below computes the sum of numbers seen so far

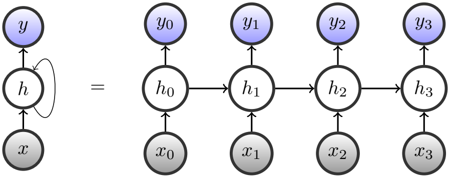


- x is the current input, h is the sum of all seen numbers, and y is the output ($= h$). Activations are identity.
- The network has been unfolded as a feedforward network with

$$h_t = x_t + h_{t-1},$$
$$y_t = h_t.$$

Example: An RNN for autoregression

- We get an autoregressive model if
  - y_t is the prediction for x_{t+1}
  - $h_t = [x_t, \ldots, x_{t-p+1}]$.



Three major classes of RNNs

Vanilla RNN

$$h_t = \tanh\left(W\begin{pmatrix}h_{t-1}\\ x_t\end{pmatrix} + b\right)$$

LSTM

$$\begin{pmatrix}f_t\\ i_t\\ \tilde{c}_t\\ o_t\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \tanh\\ \sigma\end{pmatrix}\left(W\begin{pmatrix}h_{t-1}\\ x_t\end{pmatrix} + b\right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t,$$
$$h_t = o_t \odot \tanh(c_t).$$

GRU

$$\begin{aligned}
r_t &= \sigma(W_r[h_{t-1}, x_t] + b_r),\\
\tilde{h}_t &= \tanh(W_c[r_t \odot h_{t-1}, x_t] + b_r),\\
z_t &= \sigma(W_z[h_{t-1}, x_t] + b_z),\\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t.
\end{aligned}$$

capabilities

a growing frontier

machine learning 102

maths and intuition

PDE solver

science is a differential equation

Alan Turing

discrete optimization

used by

50% Edelman Award winners

George Nemhauser

time series modelling

time is the only thing

Miles Davis

roadmap

a path forward

A Path Forward

- Lots of things not mentioned here, and more on ML/DL here.
- Many excellent courses/books for learning ML/DL.
- Learn general-purpose ML/DL tools
  - highly recommended: sklearn for ML, PyTorch for DL
  - they have excellent user interface, documentations and support communities
- Many pointers in good surveys
  - PDE: e.g., (Karniadakis et al., 2021)
  - Discrete optimization: e.g.,(Bengio, Lodi, and Prouvost, 2020)
  - Time series: e.g., (Ismail Fawaz et al., 2019; Lim and Zohren, 2021)

  Many papers provide publicly available implementations.

Call for Papers
Annals of Operations Research

Special Issue: Decision-Making Under Uncertainty: A Multidisciplinary Perspective

Submission deadline: April 15, 2023

How do we make good decisions in the presence of uncertainty? This question arises in numerous contexts, including natural resources management and robot planning and control. The past few decades have seen significant advances in decision-making under uncertainty. These range from new domain-independent methods in areas such as artificial intelligence, statistics, operations research, robot planning, and control theory, to novel domain-specific methods in fields such as ecology, fisheries, economics, and mathematical finance. Unfortunately, progress in one domain may often be easily overlooked by researchers from another community.

This special issue calls for papers that provide a multidisciplinary perspective on the theory, practice, and computational techniques for decision-making under uncertainty. Submissions should demonstrate how the work is relevant to researchers from different communities. Examples include theoretical studies of decision models relevant to disparate fields, and novel applications of tools from one field to another.

https://dmuu2022.github.io/

C*omputing and math, a powerful mix,*
A*rtificial intelligence, what a fix!*
I*nsight and accuracy, our goal in sight,*
N*ew solutions emerge, to solve and delight.*
S*cience and tech, a future so bright!*

ChatGPT

Slides: https://yenan.github.io/talks/anziam23