What mathematics in the future marked by AI

ALESSIO DRIVET
GeoGebra Institute of Turin
ITALY

Abstract: - The text addresses a significant topic: the relationship between mathematics and artificial intelligence. In particular, the focus is on the most involved issues and on which tools can be used to present the most relevant topics to current students. As an aid to understanding, a program in Python exemplifies the mathematical topics addressed.

Key-Words: - AI, Apps, Mathematics, Machine learning, Python.

Received: March 13, 2024. Revised: October 15, 2024. Accepted: November 12, 2024. Published: December 11, 2024.

1 Introduction

Predicting the future of mathematics is complex but considering the growing impact of artificial intelligence (AI) and machine learning (ML), we can identify some significant directions (MacKay, 2003; Russell & Norvig, 2016; Deisenroth et al., 2020; Romero & Ventura, 2020; Bengio et al., 2021; Luckin & Cukurova, 2022; Chiu et al., 2023).

To address this topic, we need to ask two questions:

1. What tools to work with artificial intelligence and machine learning?
2. What mathematics is appropriate for this challenge?

As a first approximation, using the Web or apps that exploit artificial intelligence is possible. AI is the ability of a computer system to source, organize, and process data by imitating human intelligence. Can solve problems and make decisions, in a sense, "reason", without direct human intervention. First of all, given the great emphasis given by media outlets to ChatGPT (Wardat et al., 2023), Gemini, Copilot, Claude, GPT4All, and other programs, it must be underlined that AI is an umbrella term that covers a wide range of technologies and applications that aim to emulate human intelligence, while Large Language Models (LLMs) are a specific class of AI designed to understand and generate natural language. While both use machine learning and enormous amounts of data, their applications, methodologies, and scopes of use differ significantly. AI includes various technologies and methodologies to create machines that imitate intelligent human behaviours: computer vision, voice recognition, and robotics. Researchers design LLMs to focus on natural language processing and to understand and generate human language coherently. From the point of view of mathematics and computer science, the former includes a range of techniques such as neural networks (García Planas, 2019), decision trees, genetic algorithms, expert systems, and other advanced computational methods. The latter are mainly based on deep neural network architectures and statistical principles to process and generate text; they are first trained on copious amounts of text using unsupervised learning techniques.

2 Tools and technologies

Several math apps use AI to assist students in solving math problems by showing step-by-step steps and solutions for various exercises. Furthermore, there are several solutions, such as *Colab* or *Amazon SageMaker*. These platforms provide potent environments for developing and deploying machine learning models. However, using them may require in-depth knowledge of programming and machine learning principles.

Here are some examples of Apps:

- *Photomath*: a standout in the world of math apps, uses AI to create an interactive learning experience. Students can instantly access step-by-step solutions by simply taking a photo of a math problem, making complex problems more manageable.
- *Microsoft Math Solver*: This App's AI includes language analysis algorithms to

understand mathematical problems written in natural language, drawn, or entered by keyboard.

- *Desmos*: The App is best known as a graphing calculator, but it increasingly integrates AI-powered features to improve user experience and analytics capabilities.
- *Mathway*: This is an AI-powered recommendation system. The system analyses the user's input and guides how to proceed. These suggestions include reviewing math concepts, steps to solve the problem, and additional resources. However, it would help if one subscribed to get step-by-step explanations.
- *Wolfram Alpha*: This knowledge engine (https://www.wolframalpha.com/) uses AI to answer questions about many subjects, including mathematics. It is beneficial for its ability to solve complex mathematical problems and provide guided solutions. The App is available for iOS and Android for a modest fee.
- *Algor Education*: This online App (https://www.algoreducation.com/it) uses AI to create automatic concept maps from texts, photos, and audio. It is not designed only for mathematics but can prove helpful in this discipline. However, one needs to make a monthly payment and pay it annually.

In addition to applications, it is crucial to know the most beneficial programming languages for machine learning and deep Learning (Goodfellow et al., 2016; Janiesch et al., 2021):

- Python (https://www.python.org/) is not only one of the most common languages (Raschka, 2015; Ramalho, 2022) for ML and DL but also one of the most user-friendly. Many libraries such as NumPy (for numerical operations), Pandas (for data manipulation and analysis), Scikit-Learn (for implementing many basic algorithms), Matplotlib (for viewing graphs and data), Itertools (for managing iterators), and other essential tools support it. These libraries can be easily loaded from the terminal with pip install xxxxx.
- R (https://www.r-project.org/) is a programming language and a versatile software environment (Wickham et al., 2023). Its applications range from statistical analysis to graph creation and visualization. R's support for numerous machine learning libraries, including algorithms like *Caret* and *RandomForest,* makes it a powerful tool for classification, regression, clustering, and supervised and unsupervised learning. One can use numerous environments and products to develop, train, and deploy models for machine learning:
- *Some Python libraries,* such as Pandas and Sklearn, allow the implementation of fundamental decision tree classifiers *(DecisionTreeClassifier).*
- *RStudio* (https://posit.co/download/rstudio-desktop/) is an IDE (Integrated Development Environment) for R that supports Python and other languages via integrations.
- *KNIME* (https://www.knime.com/) is an open-source data analytics platform that integrates machine learning and AI tools. The platform offers a wide range of nodes to run algorithms based on complex mathematical models, such as classification algorithms, regression, clustering, and other advanced analytical techniques. Thanks to its user-friendly interface, KNIME (Konstanz Information Miner) is an excellent educational tool for teaching applied mathematics and AI concepts. However, much knowledge is required for a sufficiently in-depth use.
- *Teachable Machine* is an online tool from Google that allows you to create simple models using the webcam or sample data (https://teachablemachine.withgoogle.com/) . It requires no programming; you upload images, sounds, or poses to train the model. Then, the model is validated using different data to verify its accuracy (Carney et al., 2020).
- *Machine Learning for Kids* (https://machinelearningforkids.co.uk/) is a website, conceptually similar to the previous one, that extends Scratch to teach children how to create ML projects (Lane, 2021). It is an educational resource designed to teach children basic machine-learning concepts. It is available in two versions: the online version offers free access from any web

browser, while the app version requires payment and is available for iPads and Android tablets.

3 Essential Mathematics

Understanding the mathematics behind AI is essential for anyone who wants to work in this field. It allows for the development of more effective models, the correct interpretation of results, and the innovation of new techniques. Incorporating essential mathematics into schools improves understanding of emerging technologies and stimulates critical thinking and problem-solving skills.

Here are some key areas where essential AI mathematics should impact schools and some Python programs to exemplify this:

1. *Linear Algebra*

It is a branch of mathematics that deals with linear equations and vector spaces (Strang, 2005; Boyd & Vandenberghe, 2018). It provides the basis for many machine learning algorithms, such as Principal Components Analysis (PCA).

Neural networks, machine learning, and deep learning algorithms use vector (and matrix) operations. These represent data, model weights, and operations that transform data across neural networks' various layers. Vector dot product and matrix-vector multiplication are essential building blocks for various artificial intelligence algorithms, including neural networks.

Example: Suppose we have two arrays that, in an investment decision, could represent the weights assigned to various criteria (risk, return, liquidity) and the scores of a specific option. The dot product can be calculated to measure the similarity between data or to project one vector onto another, which is helpful in many applications, such as calculating principal components.

In a neural network (Nielsen, 2015), we consider a matrix representing the weights between an input layer with m neurons and an output layer with n neurons, where each row of the matrix represents the weights connected to an output neuron. Multiplying this matrix by an input vector representing characteristic coefficients or scores provided to the model gives the output. Performing the matrix-vector product obtains a new representation of the input features. Subsequent model layers use this new representation for prediction, classification, or other processing tasks.

neuronal_connections.py

```
import numpy as np
# Two vectors
a = np.array ([0.3, 0.5, 0.2])
b = np.array ([7, 8, 6])
# Scalar product
dot_product = np.dot(a, b)
# Weight matrix
W = np.array ([[0.2, 0.8], [0.5, 0.1], [0.9, 0.3]])
#Input vector
input = np.array ([1.0, 2.0])
#Output calculation
output = np.dot(W, input)
#Printing the results
print ("Dot product: ", dot_product )
print ("Output: ", output)
```

2. *Mathematical Analysis*

Calculus has been considered the prince of mathematics since the 18th century, but now it can take on very particular characteristics (Fitzpatrick, 2009; Calonge et al., 2023). As is optimization theory, the gradient descent algorithm is central to training ML models, including stochastic optimization methods, which underlies many learning algorithms.

The gradient plays a crucial role in neural network training. We use it to adjust weights and biases dynamically, a pivotal process to the network's learning. The backpropagation algorithm, a workhorse of neural network training, calculates the gradients of the loss function with respect to each network parameter. These gradients are then harnessed to update the parameters using the gradient descent method.

The example below defines the quadratic loss function and its gradient concerning the theta parameter. This basic implementation of deterministic gradient descent is a fundamental optimization concept.

gradient.py

```
import numpy as np
import matplotlib.pyplot as plt
# Loss function
def loss_function (theta):
   return theta**2
# Gradient of the loss function
def gradient(theta):
   return 2 * theta
# Initial parameter
theta = 10.0
# Learning rate
learning_rate = 0.1
# Number of iterations
iterations = 30
# To track the evolution of the parameter
theta_values = []
loss_values = []
for _ in range(iterations):
   theta_values.append (theta)
   loss_values.append ( loss_function (theta))
# Update parameter
theta = theta - learning_rate * gradient(theta)
# Print final values
print ( f'Final value of theta: {round(theta, 4)}')
print ( f'Final value of the loss function: {round(
loss_function (theta), 4)}')
# View
plt.figure ( figsize =(10, 5))
# Graph of the loss function
theta_interval = np.linspace (-12, 12, 100)
plt.plot ( theta_interval , loss_function (
theta_interval ), label='Loss Function')
# Upgrade points
plt.scatter ( theta_values , loss_values , color='red',
label='Updates')
plt.title ('Gradient Descent Optimization')
plt.xlabel ('Theta')
plt.ylabel ('Leak')
plt.legend ()
plt.grid (True)
plt.show ()
```

3. *Probability and Statistics*

Statistics and probability provide the theoretical foundation and practical tools to build and refine effective models in various fields, including machine learning and deep learning (Hastie et al., 2009). AI relies heavily on analysing large amounts of data and making accurate inferences. Just think of Bayesian algorithms (Koller & Friedman, 2009; Gelman et al., 2020) that update parameter estimates with new evidence or Monte Carlo methods, including Markov chain Monte Carlo sampling (MCMC), which can approximate complex distributions.

This program vividly illustrates the practical application of the Monte Carlo method in AI. Here, an AI agent is the key player, using the method to make decisions under conditions of uncertainty. The process involves running simulations, after which the agent selects the action with the highest average reward.

agent.py

```
import random
class AIAgent:
   def __init__(self):
      self.actions = ['A', 'B', 'C']
      self.rewards = {'A': 0, 'B': 0, 'C': 0}
      self.counts = {'A': 0, 'B': 0, 'C': 0}
   def choose_action(self, num_simulations):
      for _ in range(num_simulations):
         action = random.choice(self.actions)
         reward = self.simulate_action(action)
         self.rewards[action] += reward
         self.counts[action] += 1
      # Choose the action with the highest average
reward
      return max(self.actions, key=lambda a:
self.rewards[a] / max(1, self.counts[a]))
   def simulate_action(self, action):
      # Base reward for all actions
      base_reward = random.normalvariate(5, 1)
      # Action-specific rewards
      if action == 'A':
         return base_reward + random.choice([0, 2]) #
50% chance of +2 bonus
      elif action == 'B':
         return base_reward + random.uniform(0, 2) #
Random bonus between 0 and 2
      else:  # Action C
         return base_reward + 1 if random.random() <
0.5 else base_reward  # 50% chance of +1 bonus
# Using the agent
agent = AIAgent()
best_action = agent.choose_action(1000)
print(f"The best action according to the AI agent is:
{best_action}")
```

```
# Print statistics
for action in agent.actions:
   avg_reward = agent.rewards[action] / max(1, agent.counts[action])
   print(f"Action {action}: average reward = {avg_reward:.2f}, count = {agent.counts[action]}")
```

ML and DL algorithms, supported by statistics and probability, are used to recognize patterns in data (Bishop, 2006). Think of speech recognition, computer vision, and natural language processing. Probability is also critical to identifying deviations from standard patterns in the data.

Consider this example: We have a dataset with emails labelled as spam or non-spam and the words in each email. We aim to create a machine-learning model that automatically categorises emails as spam or non-spam based on their word content using a Naive Bayes classifier (Lv et al., 2020; Ghada et al., 2021). Based on the observed characteristics, this classifier calculates the back probability of each class, which is the probability that the instance belongs to that class. The class with the highest posterior probability is assigned to the instance as the predicted class. This decision-making process, heavily reliant on statistics and probability, is the backbone of the model's functionality.

NaiveBayes.py

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
# Training data
X_train = [
"appointment at 11",
"to read",
"meeting on the 27th",
"discount on purchase",
"this is just an example",
"beware of the virus",
"advertising",
"appointment at 4pm",
"enter password",
"English lesson"
]
# Training labels
y_train = [0, 0, 0, 1, 0, 1, 1, 0, 1, 0]
# Test data
word_to_search = input("Enter the word to search for: ")
X_test = [ word_to_search ]
# Vectorization of training and test data
vectorizer = CountVectorizer ()
X_train_vectorized = vectorizer.fit_transform (X_train )
X_test_vectorized = vectorizer.transform (X_test)
# Naive Bayes model training
classifier = MultinomialNB ()
classifier.fit ( X_train_vectorized , y_train )
# Forecasts
predictions = classifier.predict ( X_test_vectorized )
if predictions[0] == 1:
   print("Spam")
else:
   print ("non-spam")
```

4. Theory of Algorithms

Algorithms are the foundation of programming. Understanding sorting algorithms, searching, and data structures is critical to efficiently implementing AI models. They are based on mathematical and computational principles continually improved through research (Nocedal & Wright, 1999). Some recurring terms are Supervised Learning, in which algorithms learn from a labelled data set, making predictions or classifications; Unsupervised Learning, in which algorithms find patterns or structures in unlabelled data; and Reinforcement Learning, in which Learning occurs through interaction with the environment, maximizing a cumulative reward.

Among the practical applications of AI algorithms, we can highlight: Computer Vision, in which Convolutional Networks (CNN) are used for image and video recognition; segmentation algorithms to identify and separate objects in an image; Language Models used for machine translation, text generation, or sentiment analysis; paths on graphs (West, 2001); and search algorithms used to retrieve relevant information from large corpora of text.

Here is a practical example: a keyword cloud is generated from the text file xxxxx.txt, which is located in the same directory. This process involves the use of AI algorithms to extract and visualize the most frequently occurring words in the text, providing a quick overview of its content.

word_cloud.py

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
# Function to read the content of the file
def read_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        return file.read()
# Function to filter short words
def filter_short_words(text, min_length=4):
    words = text.split()
    filtered_words = [word for word in words if
len(word) >= min_length]
    return ' '.join(filtered_words)
# Ask the user to input the file name
file_name = input("Enter the name of the text file
(in the same directory): ")
try:
    # Read the content of the file
    text = read_file(file_name)
    # Filter short words
    filtered_text = filter_short_words(text)
    # Define stopwords and filter those with three or
fewer letters
    stopwords = set(STOPWORDS)
    stopwords = {word for word in stopwords if
len(word) > 3}
    # Generate the word cloud
    wordcloud = WordCloud(
        width=800,
        height=400,
        background_color='white',
        stopwords=stopwords,
        max_words=30,  # Limit the maximum number
of words
        contour_color='steelblue',
        contour_width=2,
        min_font_size=10
    ).generate(filtered_text)
    # Display the generated image
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()
except FileNotFoundError:
    print(f"The file {file_name} was not found.
Please make sure the file exists in the same
directory as the script.")
```

5. Discrete Mathematics

There are many areas of discrete mathematics relevant to AI. Among these, we can mention the study of graph structures used to analyse connections and influences between individuals in a social network or in problems of finding the optimal path. Another fundamental chapter is combinatorics, where we address assignment and scheduling problems and consider all possible combinations.

The following program allows the generation of a project with a specified number of elements (n), grouped into subsets of fixed size (b), with the characteristic that each pair of subsets shares a specific number of common elements (e).

project.py

```
from itertools import combinations
def generate_project (n, b):
    elements = set(range(n))
    project = [set(subset) for subset in combinations
(elements, b)]
    return project
def print_project (project):
    for i, subset in enumerate(project, 1):
        print ( f"Subset {i}: {subset}")
def main ():
    n = int (input("Enter the number of project
elements: "))
    b = int (input("Enter the number of elements per
subset: "))
    e = int (input("Enter the number of common
elements between each pair of subsets: "))
    if b <= n and e < b:
        project = generate_project (n, b)
        print ( f"Generated a ({n}, {b}, {e})-valid
project:")
        print_project (project)
    else:
        print ("Unable to generate a valid project with
the supplied parameters.")
if __name__ == "__main__":
    main()
```

4 Conclusions

AI technologies cover a wide range of techniques used for developing and implementing intelligent systems. Techniques such as Supervised, Unsupervised, and Reinforcement Learning have previously been mentioned. Generative Models and Natural Language Processing are other aspects that could be explored in depth, but this would excessively expand the text. One additional point relates to the stability of an artificial intelligence

system. In this case, we refer to the ability of the model to maintain reliable and constant performance, even in the face of variations in input data or disturbances. Ensuring the stability of an AI system requires continuous attention to various aspects, from training to operational robustness (Caramanis et al., 2012; Qiu et al., 2016).

References:

[1] Bengio, Y., Lecun , Y., & Hinton, G. (2021). Deep learning for AI. *Communications of the ACM* , *64* (7), 58-65.

[2] Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer google schola*, 2, 1122-1128.

[3] Boyd, S., & Vandenberghe, L. (2018). *Introduction to applied linear algebra: vectors, matrices, and least squares* . Cambridge University Press.

[4] Calonge, D. S., Smail, L., & Kamalov, F. (2023). Enough of the chit-chat: A comparative analysis of four AI chatbots for calculus and statistics. *Journal of Applied Learning and Teaching* , *6* (2).

[5] Caramanis, C., Mannor, S., & Xu, H. (2012). 14 Robust Optimization in Machine Learning. *Optimization for Machine Learning*, 369.

[6] Carney , M., Webster, B., Alvarado, I., Phillips, K., Howell, N., Griffith, J., ... & Chen, A. (2020, April). Teachable machine: Approachable Web-based tool for exploring machine learning classification. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems* (pp. 1-8).

[7] Chiu , T. K., Xia, Q., Zhou, X., Chai, C. S., & Cheng, M. (2023). Systematic literature review on opportunities, challenges, and future research recommendations of artificial intelligence in education. *Computers and Education: Artificial Intelligence* , *4* , 100118.

[8] Deisenroth , M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for machine learning* . Cambridge University Press.

[9] Fitzpatrick, P. M. (2009). Advanced Calculus, Pure and Applied Undergraduate Texts vol. 5. *American Mathematical Society, Providence* .

[10] García Planas, M. I. (2019). Analyzing Controllability of dynamical systems modelling brain Neural Networks. *WSEAS transactions on systems*, *18*, 304-312.

[11] Gelman, A., Vehtari, A., Simpson, D., Margossian, C. C., Carpenter, B., Yao, Y., ... & Modrák, M. (2020). Bayesian workflow. *arXiv preprint arXiv:2011.01808*.

[12] Ghada, A. R., Mamat, R. B., & Rawashdeh, J. H. (2021). Evaluation of the Performance for Popular Three Classifiers on Spam Email without using FS Methods. *WSEAS Transactions on Systems and Control*, *16*, 121-132.

[13] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* . MIT press.

[14] Hastie, T., Tibshirani , R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: Springer.

[15] Janiesch , C., Zschech , P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets* , *31* (3), 685-695.

[16] Koller, D., & Friedman, N. (2009). Probabilistic graphical models: principles and techniques. MIT press.

[17] Lane, D. (2021). *Machine learning for kids: A project-based introduction to artificial intelligence* . No Starch Press.

[18] Luckin , R., George, K., & Cukurova , M. (2022). *AI for school teachers* . CRC Press.

[19] Lv, T., Yan, P., Yuan, H., & He, W. (2020, June). Spam filter based on naive Bayesian classifier. In *Journal of Physics: Conference Series* (Vol. 1575, No. 1, p. 012054). IOP Publishing.

[20] MacKay, D. J. (2003). *Information theory, inference and learning algorithms* . Cambridge University Press.

[21] Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25, pp. 15-24). San Francisco, CA, USA: Determination press.

[22] Nocedal , J., & Wright, S. J. (Eds.). (1999). *Numerical optimization* . New York, NY: Springer New York.

[23] Qiu, J., Wu, Q., Ding, G., Xu, Y., & Feng, S. (2016). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, *2016*, 1-16.

[24] Raschka, S. (2015). *Python machine learning* . Packt publishing ltd.

[25] Ramalho, L. (2022). *Fluent python* . "O'Reilly Media, Inc.".

[26] Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. *Wiley interdisciplinary reviews: Data mining and knowledge discovery* , *10* (3), e1355.

[27] Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Pearson.

[28] Strang, G. (2005). *Linear algebra and its applications* . Brooks/Cole.

[29] Wardat , Y., Tashtoush , M. A., AlAli , R., & Jarrah, A. M. (2023). ChatGPT: A revolutionary tool for teaching and learning mathematics. *Eurasia Journal of Mathematics, Science and Technology Education* , *19* (7), em2286.

[30] West, D. B. (2001). *Introduction to graph theory* (Vol. 2). Upper Saddle River: Prentice hall.

[31] Wickham, H., Çetinkaya -Rundel, M., & Grolemund , G. (2023). *R for data science* . "O'Reilly Media, Inc.".

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflict of interest to declare that is relevant to the content of this article.