# IOAI 2027 Syllabus

International Olympiad in Artificial Intelligence (IOAI)

## Introduction

The International Olympiad in Artificial Intelligence (IOAI) is a premier global competition for high school students, aiming to cultivate both a strong theoretical foundation and hands-on expertise in Artificial Intelligence. This syllabus outlines the topics contestants should master to excel in the competition. Each year, the IOAI International Scientific Committee (ISC) updates the official syllabus to reflect the latest research findings and educational priorities.

## Topic Classifications

The topics are categorized into three distinct sections, indicating the level and nature of knowledge contestants need:

1. **Theory (How it works)**: Contestants should understand core concepts and theoretical underpinnings—the "why" behind AI. This may involve studying textbooks, courses, and other resources to delve into the mechanics that power AI algorithms. Breadth should be prioritized over depth in covering all relevant topics.

2. **Practice (What it does, when to use it, and how to implement it)**: Contestants should develop practical skills necessary to implement AI methods in code. This includes knowing how to use library functions effectively, call the method on a particular data, and interpret outputs. Example: While a contestant need not fully dissect the internal workings of the Adam optimizer, they should be able to decide when and how to employ it.

3. **Both**: Certain topics require knowledge of both theoretical principles and practical application.

This structured approach ensures that contestants acquire the right balance of conceptual insight and hands-on proficiency across the diverse array of AI topics. Topics not listed are considered excluded from the syllabus. Questions can be directed to isc@ioai-official.org.

# 1 Foundational Skills & Classical Machine Learning

| Topic | Subtopic | Category |
|---|---|---|
| Programming Fundamentals | Python Basics (Loops, Functions, etc.) | Practice |
| | NumPy and Pandas for Data Handling | Practice |
| | Matplotlib and Seaborn for Visualization | Practice |
| | Scikit-learn for ML | Practice |
| | PyTorch Basics | Practice |
| | Tensor (Multi-dimensional Array) Manipulation | Practice |
| | Training Models on CPU and GPU | Practice |
| Supervised Learning | Linear Regression | Both |
| | Logistic Regression | Both |
| | L1 & L2 Regularization | Both |
| | K-Nearest Neighbors (K-NN) | Both |
| | Decision Trees | Both |
| | Model Ensembles (Gradient Boosting, Bagging, Random Forest) | Practice |
| | Support Vector Machines (SVM) | Both |
| Unsupervised Learning | K-Means Clustering | Both |
| | Principal Component Analysis (PCA) | Both |
| | t-SNE, UMAP | Practice |
| | DBSCAN, Hierarchical & Spectral Clustering | Practice |
| Data Science Fundamentals | Model Evaluation Metrics (Accuracy, Precision, Recall, F1-Score, etc.) | Both |
| | Underfitting, Overfitting | Theory |
| | Hyperparameter Tuning | Practice |
| | Cross-Validation | Practice |
| | Confusion Matrix and ROC Curves | Both |
| | Feature Engineering * | Practice |
| | Data Processing ** | Practice |
| Technical Fundamentals | Basic Linux terminal commands *** | Practice |
| | Jupyter Notebook / JupyterLab | Practice |
| | Basic Git **** | Practice |
| | Work on remote servers (via SSH) | Practice |

* Feature Engineering involves transforming raw, potentially high-dimensional data, categorical data, time series, or ragged data into a compact set of informative features. Techniques involve sliding windows, pooling operations, one-hot encoding, statistical moment-based features (average, standard deviation), PCA and neural-network-based embeddings.

** Data Processing concerns the handling of missing data and irregular data, including in sequence modeling settings. Techniques involve basic imputation strategies (mean/median/forward-fill for sequences) and padding for variable-length sequences. Covered here are also normalization and standardization techniques, train/validation/test splitting strategies, basic data augmentation (flipping, cropping, noise addition), tokenization and vocabulary building for text and audio and patching for images. The data can be text, tabular, image, audio, video, and time-series, and should be processed with the methods listed in this document.

*** Commands include: ls / cd / rm / mkdir / chmod / sudo / nvidia-smi / htop / ps / grep

**** Commands include: add / commit / push / reset / diff / status / restore

## 2 Neural Networks & Deep Learning

| Topic | Subtopic | Category |
|---|---|---|
| Neural Networks | Perceptron Basics | Both |
| | Gradient Descent | Both |
| | Backpropagation | Both |
| | Activation Functions (ReLU, Sigmoid, Tanh) | Both |
| | Loss Functions (MSE, MAE, Cross Entropy, etc.) | Both |
| Deep Learning | Multi-Layer Perceptrons (MLP) | Both |
| | Data Embeddings (text, image, audio) | Both |
| | Pooling Techniques (Max, Average) | Both |
| | Attention Mechanism | Both |
| | Transformers (theory needed only for text and image) | Both |
| | Autoencoders | Practice |
| | SGD, Mini-Batch Gradient Descent | Both |
| | Momentum Methods (Adam, AdamW) | Practice |
| | Convergence and Learning Rates | Practice |
| | Regularization: Dropout, Early Stopping, Weight Decay | Practice |
| | Weight Initialization | Practice |
| | Batch Normalization | Practice |
| | Full and Parameter-efficient Finetuning Methods | Practice |

## 3 Computer Vision

| Topic | Subtopic | Category |
|---|---|---|
| Fundamentals | Convolutional Layers | Both |
| | Image Classification | Practice |
| | Object Detection (YOLO, SSD, DETR) | Practice |
| | Image Segmentation (U-Net) | Practice |
| | Pre-trained Vision Encoders (e.g. ResNet) | Practice |
| | Image Augmentation Techniques | Practice |
| | Generating Images with GANs | Practice |
| | Self-Supervised Learning for Vision | Practice |
| | Vision-text encoders (e.g. CLIP) | Practice |
| | Diffusion Models | Practice |

## 4 Natural Language Processing & Audio

| Topic | Subtopic | Category |
|---|---|---|
| NLP | Text Classification | Practice |
| | Sequence-to-Sequence tasks | Practice |
| | Pre-trained Text Encoders | Both |
| | Language Modeling | Both |
| | Text Decoders (conditional generation) | Practice |
| | Pre-trained Language Models: Open-source and Closed-source (API-based) | Practice |
| Audio Processing | Pre-trained Audio Encoders: HuBERT | Practice |
| | Audio Models: Qwen-Audio, Whisper, Voxtral | Practice |

# 5 Basics of Reinforcement Learning

| Topic | Subtopic | Category |
|---|---|---|
| Reinforcement Learning * | Agent–Environment Loop (States, Actions, Rewards, Episodes) | Both |
| | Policies, Returns and the Discount Factor | Both |
| | State-Value and Action-Value Functions (V, Q) | Both |
| | Q-learning, DQN, REINFORCE | Practice |

* IOAI 2027 will gently introduce the students to the basics of Reinforcement Learning. We recommend using the following textbook as a reference: Sutton & Barto, *Reinforcement Learning: An Introduction (2nd ed.)* [link], with a focus on:

- **MDPs and Dynamic Programming**
  - States, actions, rewards, transitions, returns, and episodes (Sections 3.1–3.3)
  - Policies, value functions, Bellman equations, and optimality (Sections 3.5–3.6)
  - Policy evaluation and policy improvement (Sections 4.1–4.2)
- **Value-Based Learning: Q-Learning and DQN**
  - Temporal-difference prediction and bootstrapping (Section 6.1)
  - Q-learning, off-policy learning, and $\varepsilon$-greedy exploration (Section 6.5)
  - Value-function approximation, prediction objectives, and semi-gradient updates (Sections 9.1–9.3 and 11.1)
  - Deep Q-networks, experience replay, and target networks (*Human-level control through deep reinforcement learning*, Mnih et al., 2015)
- **Policy-Based Learning: REINFORCE**
  - Monte Carlo estimation of state and action values (Sections 5.1–5.2)
  - Parameterized policies and their advantages (Section 13.1)
  - REINFORCE: Monte Carlo policy gradients (Section 13.3)