

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS EDUCATIONAL
INSTITUTION OF HIGHER EDUCATION
"NORTH CAUCASUS FEDERAL UNIVERSITY"

SUGGESTIONS FOR TEACHING
for the performance of practical work
in the discipline "Artificial intelligence in the professional sphere"
for students of the field of study
11.03.02 Infocommunication Technologies and Communication Systems
Profile
"Intelligent Data Processing in Infocommunication Systems and Networks"

Table of Contents

Introduction

Practical Work No1 The Basics of Working with Jupyter Notebook, JupyterLab, and Google Colab

Practical Work No2 NumPy Package Basics

Practical Work No3 Basics of working with the matplotlib package

Practical Work No4 Introduction to pandas: Object Series

Practical Work No5 Introduction to pandas: DataFrame object

Practical Work No6 Fundamentals of Research Data Analysis

Practice No7 Linear Regression

Practical Work No8 Logistic Regression

Practice No9 Quality Metrics of Classification Models

Practical Paper No10 Nearest Neighbor Method and Support Vector Method

Practical Work No11 Decision Trees

Practical Work No12 Random Forest

Practical Work No13 Gradient Boosting

Practical Work No14 Unsupervised Learning: Data Clustering Tasks

Practical Work No15 Unsupervised Learning: Data Dimensionality Reduction Problems

Practical Work No16 Selection of Model Hyperparameters

Conclusion

Educational, methodological and information support of the discipline

INTRODUCTION

1. Purpose and objectives of mastering the discipline (module)

The purpose of the discipline is to form, according to the educational program, the competencies of the UK-1 student.

To achieve this goal, the following tasks are solved in the process of studying the discipline: familiarization with the basic concepts, history of the formation and directions of development of artificial intelligence technology; study of methods of uninformed and informed search in state space, including implementation in Python using the NetworkX library; mastering the methods of local search and genetic algorithms, understanding their application in various tasks; mastering the principles of adversarial search, including minimax and alpha-beta clipping algorithms; acquaintance with the concept of intelligent agents and inference based on the logic of statements; study of approaches to natural language processing, including working with transformers and large language models using the transformers library and Ollama, YandexGPT or GigaChat platforms; mastering the basics of reinforcement learning and obtaining skills in building agent-based models with elements of adaptive behavior; studying the basics of computer vision, including image processing, object recognition and three-dimensional scene reconstruction; developing the ability to apply the learned AI methods as part of project and research activities in the professional field; formation of skills for analysis, interpretation and presentation of results, including the formulation of solutions and their presentation in the form of presentations and public speeches.

2. Place of the discipline (module) in the structure of the educational program

The discipline belongs to optional disciplines.

3. List of planned learning outcomes in the discipline (module) correlated with the planned results of the development of the educational program

Code, formulation of competence	Indicator code, wording	Planned learning outcomes in the discipline (module), characterizing the stages of the formation of competencies, indicators
UK-1. Able to search, critically analyze and synthesize information, apply a systematic approach to solving problems	ID-1 UK-1 identifies a problem situation, carries out its multifactorial analysis and diagnosis based on a systematic approach;	Analysis of the implementation of requirements; Development of options for the implementation of requirements; assessment and justification of recommended solutions; selection of means for implementing software requirements; use of existing standard solutions and software design templates; development of technical specifications for software components and their interaction.

	ID-2 UK-1 searches, selects and systematizes information to determine alternative options for strategic solutions in a problem situation;	Understanding of the methods of classical system analysis; Understanding of the theory of business process management; Understanding of conceptual design methods; Understanding of methods for assessing the quality of software systems.
	ID-3 UK-1 determines and assesses the risks of possible solutions to a problem situation, chooses the best option for its solution.	Ability to schedule control activities; develop business requirements for the system; be able to identify key features and constraints of the system; identify subsystems of the system

4. Scope of the academic discipline (module) and forms of control *

Scope of classes: total: 3 EC 108 academic hours.	OFO, In acad. hours
Contact work:	
Lectures/from them practical training	
Laboratory work/from them practical training	36.00
Practical training/of which practical training	
Independent work	54.00
Forms of control	
Credit with a grade	Yes

5. The Importance of Students' Lab Work

Practical work is a key component of the educational process in the discipline "*Artificial Intelligence in the Professional Field*", since it provides a practical development of the methods and tools studied in the theoretical part of the course. It contributes to the formation of applied skills necessary for the design and implementation of intelligent solutions in real professional tasks.

Through practical tasks, students master the basic algorithms of artificial intelligence, gain experience with the Python programming language and specialized libraries (for example, NetworkX, transformers, OpenCV) , learn to build and analyze intelligent agents, implement methods for searching, processing texts and images, apply large language models and conduct reinforcement learning.

Practical implementation of laboratory tasks develops skills in programming, system analysis, critical thinking and independent problem solving. Students learn to make informed technical decisions, correctly interpret the results of models and evaluate the effectiveness of methods, taking into account the characteristics of the professional field.

Thus, laboratory work provides a close connection between theory and practice and is the basis for the formation of professional competence in the field of artificial intelligence, contributing to the successful application of AI tools in specific subject areas.

PRACTICAL WORK №1

Topic: Working with the NetworkX package of the Python programming language

Purpose of the work

Mastering the skills of building, visualizing and analyzing graph structures using the NetworkX library in the Python programming language. Studying methods for representing states and transitions in search problems.

Theoretical substantiation

In many artificial intelligence tasks, objects and relations between them can be represented in the form of graphs. The graph structure is used in solving problems of searching in state space, building routes, modeling network interactions, analyzing connections, etc.

The NetworkX package is one of the most common tools for working with graphs in the Python language. It provides extensive capabilities for creating both directed and undirected graphs, adding and removing nodes and edges, storing attributes of graph elements, as well as implementing algorithms for search, centrality, shortest paths, and other analysis methods.

Working with graphs involves both software implementation and visualization of the graph structure, which is especially important when interpreting results in the professional field. In artificial intelligence tasks, a graph can be a space of states, where vertices are states, and edges are possible actions. In practice, graph structures are used, for example, in planning the actions of an intelligent agent, in recommendation systems, logistics, bioinformatics, as well as in modeling the behavior of users and network systems.

Understanding basic graph operations and being able to put them into practice is the foundation for further exploration of search and decision-making techniques, including informed and uninformed methods.

Equipment and materials

A personal computer with Windows, macOS, or Linux installed

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab, or similar

Installed packages: networkx, matplotlib (for visualization)

Methodical Instructions for Laboratory Work

Safety Requirements

Work at the computer should be done in a well-lit room

It is recommended to follow the regime of working with the computer: no more than 2 academic hours without a break

The workplace must be organized in accordance with sanitary standards

The software may not be installed or modified without the permission of the instructor

You should avoid working with personal and confidential data in a learning environment

Typical tasks

1. Implement an undirected graph with five nodes and multiple edges. Display graph with labels
2. Construct a directed graph and compute the shortest path between two given vertices using Dijkstra's algorithm
3. Add edge weights to the graph and implement traversal of the graph in width, displaying the sequence of visited vertices

Control questions

1. What is a graph in the context of AI and what tasks can it represent
2. What is the difference between directed and undirected graphs
3. What types of data are used to store graphs in NetworkX

4. How NetworkX adds and removes nodes and edges
5. What are the benefits of graph visualization?
6. What graph traversal algorithms are implemented in NetworkX
7. What is a state space and how is it related to a graph?
8. How to: Implement weighted edges and use them in a search algorithm
9. What are the limitations of standard graph search algorithms?
10. How can you save and load a graph from a file

PRACTICAL WORK No2

Topic: Implementing Breadth Search with NetworkX

Purpose of the work

Familiarization with the implementation of the Breadth-First Search (BFS) algorithm in the NetworkX library. Acquisition of skills in modeling statespace search problems using graphs.

Theoretical substantiation

Breadth-first search is a basic uninformed search algorithm that is used in the problems of finding the shortest path in an unweighted graph. The algorithm starts from the initial node and sequentially examines all its neighboring vertices, then moves on to second-level neighbors, and so on until it reaches the target vertex or explores the entire graph.

BFS guarantees finding the shortest path (by the number of edges) in an undirected graph without negative weights. It is widely used in artificial intelligence systems for searching in state space, generating strategies, analyzing routes, queuing, and scheduling. The use of breadth-first search is also relevant in tasks such as the social graph, calculating the distance between elements, finding connectivity components, and verifying systems.

The NetworkX library provides convenient tools for graph modeling and BFS execution. The `networkx.bfs_edges()` and `networkx.shortest_path()` methods allow you not only to traverse the graph, but also to extract the path to the target node.

Understanding the logic of broad-first search and applying it through practical implementation is an important step in mastering intelligent algorithms and adapting them to professional tasks.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab, or similar

Installed packages: `networkx`, `matplotlib`

Methodical instructions for laboratory work

Safety Requirements

Work should be carried out in a well-ventilated and lit room

It is necessary to maintain a safe distance from the monitor (at least 50 cm)

It is recommended to take a break every 45-60 minutes of work

It is forbidden to install third-party software without the approval of the teacher

You should only work with tutorial files and public data

Typical tasks

1. Implement a graph of 7 nodes and build a visual representation of its structure
2. Find the path from a given start node to the target node using the `networkx.shortest_path()` function
3. Create a crawl queue and display the order of visiting nodes using `networkx.bfs_edges()`

Control questions

1. What is the basic idea of a breadth search
2. When BFS is More Efficient Than Other Algorithms
3. What type of data structure is used to store the node queue in BFS
4. What is the difference between BFS and DFS in terms of how it works

5. Which tasks from the professional field can be modeled using BFS
6. How the graph is implemented in the NetworkX library
7. How to Preserve the Node Traversal Order with BFS
8. How graph visualization is performed in NetworkX
9. How to check the reachability of a node using breadth-first search
10. What limitations does BFS have when applied to large graphs?

PRACTICAL WORK №3

Topic: Implementing Depth Search with NetworkX

Purpose of the work

Familiarize yourself with the implementation of the Depth-First Search (DFS) algorithm on graphs using the NetworkX library. Learn how to traverse the graph in depth and apply it to search and analysis tasks.

Theoretical substantiation

Depth search is one of the main methods of uninformed search, which implements the strategy of exploring the deepest nested paths in the graph until a goal or dead end is reached. After that, a return occurs and the search continues through other branches.

DFS does not guarantee finding the shortest path, but it is memory-efficient, as it requires storing only the current path. The algorithm is used in the tasks of traversing the decision tree, generating combinations, checking graph connectivity, planning routes, and building topological sorting.

The NetworkX library provides functions for implementing DFS, including `dfs_edges()`, `dfs_preorder_nodes()`, and `dfs_successors()`. These tools allow you to both visualize the traversal order and analyze the graph connectivity structure.

Understanding the mechanism of searching in depth and the ability to apply it in professional tasks allows students to solve a wide range of problems, including the problems of analyzing the state of systems, routing and building solutions based on a tree of options.

Equipment and materials

A personal computer with Windows, macOS, or Linux operating systems

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab, or similar

Installed packages: `networkx`, `matplotlib`

Methodical instructions for laboratory work

Safety Requirements

Work in a well-lit room, observing sanitary standards

Monitor the position of the back and the distance to the screen (at least 50 cm)

Use only teacher-authorized development tools

Comply with the regulations for the duration of work at the PC

Don't use personal data or network resources in your code without permission

Typical tasks

1. Construct a directed graph with at least 6 vertices. Visualize its structure
2. Implement graph depth traversal using `networkx.dfs_edges()` and infer the traversal order
3. Find reachable vertices from a given node and build a traversal tree

Control questions

1. What is the principle of the depth search algorithm?
2. What data structures are used in the implementation of DFS
3. In which situations DFS is preferable to BFS
4. What are the features of applying DFS to directed and undirected graphs?
5. What NetworkX features are used to implement DFS
6. What is a traversal tree and how to build one

7. How can you tell if a graph contains loops using DFS
8. What are the limitations of the DFS algorithm?
9. Can DFS be used to search infinite graphs?
10. What is the difference between `dfs_edges()` and `dfs_preorder_nodes()`

PRACTICAL WORK №4

Topic: Implementing Iterative Search with NetworkX

Purpose of the work

Learn the principles and implement Iterative Deepening Search (IDS) using Python and the NetworkX library. Learn how to combine the benefits of depth-first search and cross-breadth search.

Theoretical substantiation

Iterative deepening search is a statespace search strategy that combines the virtues of depth-first and wide-width searches. The algorithm performs a series of depth-in-depth searches, limiting them to the maximum depth, which is incremented in stages. This approach saves memory, as in DFS, while still ensuring that the shortest path is found, as in BFS.

The IDS algorithm is especially effective when there is no information about the size or structure of the state space. It is used in the problems of finding solutions in trees and graphs, planning actions, finding solutions in infinite depth spaces with a final solution.

The NetworkX library does not have a built-in IDS implementation, but the recursive DFS implementation's custom traversal and depth limitation capabilities make it possible to model this algorithm manually.

Mastering iterative search allows students to better understand strategic approaches to solving problems in the state space and learn how to adapt algorithms to the specifics of the professional subject area.

Equipment and materials

Personal computer with OS installed (Windows, macOS, Linux)

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab

Installed packages: `networkx`, `matplotlib`

Methodical instructions for laboratory work

Safety Requirements

Work in a clean and well-lit room

Observe the time limit for continuous work (no more than 45-60 minutes)

The workplace must meet the requirements of ergonomics

When performing work, use only permitted software

Do not connect third-party devices or use external resources without consent

Typical tasks

1. Implement depth-constrained depth search (DFS) in a graph using a recursive function
2. Implement iterative deepening based on bounded DFS. Set maximum depth to 5
3. Build a graph of 8-10 nodes, apply IDS, and infer the path to the target vertex if one is found

Control questions

1. What is the principle of iterative deepening search?
2. What advantages does IDS have over DFS and BFS?
3. What happens when a predetermined depth limit is reached
4. What limitations does IDS have?
5. How you can implement depth limitation when crawling in NetworkX
6. Why IDS Guarantees Shortcut Finding
7. How the graph structure affects the efficiency of IDS

8. Is it possible to use IDS in graphs with loops?
9. How IDS and search strategies are related to resource constraints
10. In what professional tasks can IDS be used?

PRACTICAL WORK №5

Topic: Implementing Greedy Heuristic Search with NetworkX

Purpose of the work

Mastering the skills of implementing greedy heuristic search on graphs using the NetworkX library. Studying the use of heuristic estimates in informed search algorithms.

Theoretical substantiation

Greedy Best-First Search is an informed search strategy that selects for expansion the vertex at which the heuristic estimate is minimal. The heuristic function ($h(n)$) approximates the distance from the current vertex to the target vertex.

Greedy search is more efficient than uninformed algorithms when there are heuristics that reflect geometric or logical proximity to the goal. However, it does not guarantee finding the shortest path, as it only takes into account the perceived proximity, not the actual cost of the path.

In professional applications, greedy search can be used for approximate route planning, prioritization of actions, and search for solutions in systems where speed is more important than optimality.

In the NetworkX library, you can implement greedy manual searches using `heapq` (priority queue) data structures and a user-defined $h(n)$ function defined as a dictionary with heuristic estimates for each vertex.

Mastering this strategy contributes to the development of an understanding of the applicability of heuristics and the formation of engineering thinking in the design of AI systems.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab, or similar

Installed packages: `networkx`, `matplotlib`, `heapq` (included in the Python standard library)

Methodical instructions for laboratory work

Safety Requirements

Work must be carried out behind serviceable equipment in accordance with sanitary standards

It is not allowed to work with confidential data

During the task, observe the time limit and take breaks

Do not change software configurations without the permission of the instructor

Only trusted sources of heuristic information should be used

Typical tasks

1. Build a weighted directed graph of 8–10 nodes. Assign a heuristic score to each vertex (as a dictionary)
2. Implement a greedy search algorithm in which a vertex with a minimum $h(n)$ is selected at each step
3. Output the traversal order and path from the start to the target vertex. Compare to BFS result

Control questions

1. What is a greedy heuristic search and how does it differ from an uninformed one?
2. How the heuristic function is given and what it reflects
3. In what cases a greedy search may not find the optimal solution
4. What data structures are used in the implementation of greedy search
5. How to store and use heuristics in NetworkX
6. What types of tasks in the professional field are suitable for greedy search

7. What are the potential dangers of overestimating or underestimating $h(n)$?
8. Why it is important to choose the right heuristics when designing a system
9. What happens when heuristic scores are equal for several nodes
10. Is it possible to combine greedy search with other strategies?

PRACTICAL WORK No6

Topic: Implementing the A* algorithm with NetworkX

Purpose of the work

Mastering the principles of the A* algorithm and its implementation using Python using the NetworkX library. Developing skills in designing heuristic functions and finding optimal paths in graphs.

Theoretical substantiation

The A* (A-star) algorithm is one of the most efficient and popular algorithms for informed search in state space. It uses two functions: $g(n)$ is the actual cost of the path from the initial vertex to the current vertex, and $h(n)$ is a heuristic estimate of the cost of the path from the current vertex to the goal.

The algorithm guarantees the optimality of the found path, provided that the heuristic is valid (does not overestimate the distance to the target). A* is used in robotics, logistics, intelligent control systems, game AI, navigation systems, and other professional tasks where it is necessary to find a path with the lowest cost.

The NetworkX library includes a built-in implementation of `networkx.astar_path()` and `networkx.astar_path_length()` that allow you to use a custom heuristic function and edge weighting parameters. You can also implement A* manually using priority queues.

Knowledge and practical application of the A* algorithm forms students' ability to build effective solutions to search problems in real professional scenarios.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab, or similar

Installed packages: `networkx`, `matplotlib`

Methodical instructions for laboratory work

Safety Requirements

Work in a well-ventilated and lit area

Comply with the standards for the duration of continuous work at the computer

Use only official libraries and APPROVED BYd development tools

Exclude the use of malicious scripts, network access without approval

Save work in a timely manner to avoid data loss

Typical tasks

1. Build a weighted directed graph of 6–8 nodes, visualize it
2. Implement the A* algorithm using `networkx.astar_path()`, specifying a heuristic function
3. Compare the A* result with the results of greedy searches and breadth-first searches. Estimate the length of the route and the number of nodes visited

Control questions

1. What is the $f(n)$ evaluation formula in the A* algorithm
2. What does a valid heuristic mean and why is it important?
3. In which cases A* may not work optimally
4. How A* differs from a greedy heuristic search
5. How to Implement Your Own Heuristic in NetworkX
6. How edge weights are set in the NetworkX graph
7. How Heuristic Accuracy Affects A* Performance

8. What professional tasks is A* used for?
9. What happens if $h(n)$ is always zero
10. What Ways to Optimize A* Implementation Do You Know

PRACTICAL WORK No7

Topic: Admissible heuristics. Dominance of heuristics. Relaxation of heuristics

Purpose of the work

Study of the concepts of admissibility and dominance of heuristic functions, mastering the principles of constructing heuristics based on the relaxation of the initial task. Acquisition of skills in analyzing the correctness of heuristic estimates in search algorithms.

Theoretical substantiation

A heuristic function is an approximation of the distance from the current vertex to the target vertex. In informed search algorithms such as A*, heuristics play a key role in the efficiency and correctness of the search.

An admissible heuristic is a heuristic that never overestimates the true cost of the path to the goal: $h(n) \leq h^*(n)$. Using a valid heuristic ensures that A* finds the shortest path.

Dominance of heuristics means that one heuristic is more informative than another if it always gives higher scores without violating admissibility. A more dominant heuristic can significantly speed up the search.

Heuristic relaxation is a method of constructing heuristic functions by simplifying (weakening) the original task. For example, in a planning task, you can temporarily ignore constraints or costs in order to get an approximate but acceptable estimate.

Practical knowledge of these concepts allows you to analyze and design heuristics adapted to specific tasks, which is especially important in the professional field - when developing navigation systems, planners and decision support systems.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab

Installed packages: `networkx`, `matplotlib`

Methodical instructions for laboratory work

Safety Requirements

Work at the computer should be carried out in comfortable conditions in compliance with sanitary standards

It is recommended to take a break every 45-60 minutes

The use of unauthorized software is not allowed

The results of the work must be saved regularly

You should only work with educational data or open sources

Typical tasks

1. Construct a graph with 8 vertices and specify two heuristics: valid and invalid. Check the correctness of the path found by A*
2. Form two heuristics, one of which dominates the other. Compare search performance
3. On the basis of the initial task, compose a relaxed problem and derive the heuristics obtained from its solution

Control questions

1. What does the admissibility of heuristics mean?
2. How to check if a heuristic is valid
3. What is the meaning of the dominance of one heuristic over another?
4. What are the advantages of using more informative heuristics?
5. What is task relaxation in the context of heuristics
6. What problem can be used to build heuristics in a routing problem

7. Why it is important not to overestimate the cost in heuristics
8. Is it possible to use an invalid heuristic in A*
9. What is the relationship between heuristics and search time
10. Give examples of relaxed tasks from the professional sphere

PRACTICAL WORK No8

Topic: Implementation of the "Hill Climbing" algorithm

Purpose of the work

Study the principles of local search and implement the Hill Climbing algorithm to optimize functions. Form an idea of the applicability of local methods in AI tasks.

Theoretical substantiation

Local search methods are used in situations where the decision space is large and there is no possibility of complete search. The "Hill Climbing" algorithm is a greedy strategy and consists in moving from the current state to the neighboring state with a higher score for the objective function. The process is repeated until the local maximum is reached, after which the algorithm stops.

The main drawback of the algorithm is the high probability of getting stuck in local maxima. Despite this, it is effective in problems where approximate solutions are permissible or the calculation time is limited.

Hill climbing can be used in the tasks of parameter setting, optimization of production processes, resource layout, routing, as well as in solving professional problems that require a quick search for a suitable solution.

The implementation of the algorithm requires defining a representation of the state, an evaluation function, and a way to generate adjacent states. When working with functions of two variables, it is convenient to visualize the progress of the search on the surface of values.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab

Installed packages: matplotlib, numpy, random

Methodical instructions for laboratory work

Safety Requirements

Work should be carried out in comfortable and safe conditions

It is necessary to avoid prolonged eye strain and observe working pauses

When using laptops, place the screen at eye level

Only trusted software is allowed to be used

Unauthorized network access or external scripts are not allowed

Typical tasks

1. Implement the "Hill Climbing" algorithm for the function $f(x)=x \cdot \sin(x)$ on the interval $[0, 10]$
2. Display the graph of the function and the trajectory of the algorithm's movement towards the local high
3. To study the impact of choosing a starting point on the search result and the number of steps

Control questions

1. How the "Hill Climbing" algorithm works
2. What is a local maximum and how does it differ from a global maximum?
3. What are the shortcomings of the algorithm and how can they be compensated for?
4. In what situations is the algorithm effective?
5. Choose a scoring feature for a professional task

6. How Step Size Affects Search Speed and Accuracy
7. What happens if there is no improvement in neighboring conditions
8. Can I use random starting points to improve search quality?
9. What is the difference between climbing a hill and simulating annealing?
10. Give examples of using local search in the professional field

PRACTICAL WORK No9

Topic: Implementation of the annealing simulation algorithm

Purpose of the work

Mastering the methods of stochastic optimization through the implementation of the simulated annealing algorithm (Simulated Annealing). Studying the mechanisms of decision-making under conditions of local extremes and random oscillations.

Theoretical substantiation

The annealing simulation algorithm refers to local search methods in which it is possible to randomly make less profitable decisions with a certain probability. This avoids getting stuck in local extremes. The principle is based on an analogy with the process of physical annealing of metals, when the heated material gradually cools down, reaching a minimum internal energy. The key components of the algorithm are: the energy function (or objective function), the initial temperature, the cooling mechanism (temperature schedule), and the rule for accepting new states. At the initial stages, the algorithm tends to explore space, and towards the end, it tends to stabilize and choose the best solution found.

Annealing simulation is used in problems where it is necessary to find an approximate optimal solution: routing, planning, resource allocation, configuration of model parameters. In the professional field, this approach is especially useful where the solution space is complex, and an exact search is impossible.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab

Installed packages: numpy, matplotlib, random

Methodical instructions for laboratory work

Safety Requirements

Work must be carried out behind equipment that meets ergonomic and sanitation standards

It is necessary to observe the work and rest regime

It is recommended to use only authorized software tools and training data

Long-term uninterrupted use of screen devices should be avoided

Intermediate results should be retained to avoid data loss

Typical tasks

1. Implement an annealing simulation algorithm for the function $f(x)=x \cdot \sin(x)$ on the interval $[0, 10]$
2. Visualize the trajectory of movement by function and analyze how the choice changes with temperature changes
3. To compare the behavior of the algorithm under different cooling schemes: linear and exponential

Control questions

1. What is the principle of simulated annealing?
2. What is the energy function and how does it relate to the objective function?
3. How is the probability of taking a worsening condition determined?
4. How temperature affects the behavior of the algorithm
5. What is the difference between simulated annealing and climbing a hill
6. What cooling schemes exist and how they affect the result

7. What is global and local low in the context of optimization?
8. In which problems is the stochastic approach preferable?
9. How to visually track algorithm behavior
10. Give examples of the use of simulated annealing in professional activities

PRACTICAL WORK №10

Topic: Implementation of Ray Search

Purpose of the work

Study of the principles of beam search and its implementation in the problems of finding solutions. Mastering the methods of reducing the search space with the help of heuristic constraints.

Theoretical substantiation

Beam search is a type of informed search based on a modification of the greedy approach. Unlike the A* algorithm or greedy heuristic search, beam search does not store all possible paths at each level, but only a limited number of the most promising ones (according to heuristics). This allows you to significantly reduce memory and time requirements, especially in tasks with wide branching.

The main parameter of the algorithm is beam width, which determines the number of alternatives stored at each level. Too small a width can lead to a loss of optimal solution, while a large width makes the algorithm closer to a full search.

Ray search is used in machine translation systems, text generation, planning, logistics, speech recognition, etc.

In Python and the NetworkX library, you can implement ray search manually by combining data structures with a custom scoring function.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab

Installed packages: `networkx`, `heapq`, `matplotlib`

Methodical instructions for laboratory work

Safety Requirements

Work in an organized environment with screen placement and lighting requirements

Avoid excessive eye strain and take regular breaks

Do not use unauthorized resources and external libraries

Save intermediate results and comment on code

Work only with training or open data

Typical tasks

1. Implement a graph with 10 vertices and assign each vertex a heuristic (e.g., distance to target)
2. Implement a beam search with a beamwidth of 2 and 3. Fix the path to the target and the number of nodes viewed
3. Compare the results of the ray search with the result of A* and the greedy search on the same graph

Control questions

1. What is the essence of the ray search algorithm?
2. What Beam Width Means and How It Affects Results
3. How the list of candidates is formed at each level
4. In what cases does ray search give an inaccurate solution?
5. How is ray search different from greedy search
6. How to Implement Priority Queue in Python
7. How nodes are selected for expansion at the next level

8. Is it possible to adapt the ray search to the tasks of text generation?
9. Give an example of the use of ray search in a professional field
10. Why Beam Search Doesn't Guarantee Finding the Optimal Solution

PRACTICAL WORK №11

Topic: Implementation of a genetic algorithm

Purpose of the work

Mastering the basics of evolutionary search through the implementation of a genetic algorithm. Formation of skills in modeling and setting up population optimization methods for solving applied problems.

Theoretical substantiation

The genetic algorithm (GA) belongs to the class of stochastic optimization methods inspired by the principles of natural selection and evolution. It operates with a population of possible solutions (individuals) that undergo the operations of selection, crossover (crossing) and mutation. Selection is carried out on the basis of the values of the objective function (fitness function).

GA does not require smoothness or continuity of the function and can be used in discrete and combined spaces. The algorithm copes well with multimodal functions and is applicable in problems where traditional methods are inefficient.

In the professional field, GA is used for scheduling tasks, resource allocation, system design, logistics, hyperparameter tuning, etc.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Python interpreter version 3.7 and higher

Development environment: Jupyter Notebook, Google Colab

Installed packages: `numpy`, `matplotlib`, `random`

Methodical instructions for laboratory work

Safety Requirements

Work in a room with sufficient lighting, behind equipment that meets the requirements of ergonomics

Observe the rules of rational labor organization, avoid long-term continuous work

Use only allowed software and training data

Regularly save results and monitor the correctness of the code

Do not use algorithms and libraries from external unverified sources

Typical tasks

1. Implement GA to optimize the function $f(x)=x \cdot \sin(x)$ on the interval $[0, 10]$ with real coding
2. Display the evolution of fitness values by generation
3. To study the influence of algorithm parameters (population size, mutation probability) on the result

Control questions

1. What is the Basic Idea of the Genetic Algorithm
2. What is an individual, population and fitness function
3. How crossover and mutation operations are implemented
4. What is the difference between a single-point and two-point crossover
5. What shutdown criteria are used in the Civil Aviation
6. What are the advantages and limitations of GA compared to gradient methods
7. Why is population diversification important?
8. How to adapt GA to tasks with constraints
9. Give examples of tasks solved with the help of civil aviation in the professional field
10. How you can visualize the process of evolution of decisions

PRACTICAL WORK №12

Topic: Working with the Python Programming Language Transformer Package

Purpose of the work

Mastering the basic skills of working with the `transformers` library to solve natural language processing problems. Familiarization with pre-trained language models and their application in professional tasks.

Theoretical substantiation

The `transformers` library, developed by Hugging Face, provides unified access to a wide range of modern deep learning models for natural language processing (NLP). It includes BERT, GPT, RoBERTa, DistilBERT, T5, and other models.

Transformers can be used to solve a variety of application problems: text classification, named entity extraction (NER), question answering (QA), summarization, text generation, and translation. The library provides convenient loading of pretrained models and tokenizers, as well as the ability to configure and retrain.

In the professional field, transformers are used to analyze documents, automatically process customer requests, intelligent search, build chatbots and voice assistants, which requires confident mastery of `transformers` tools.

Working with this library builds competencies related to the application of large language models (LLMs) in specific subject areas.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Python interpreter version 3.8 and above

Development environment: Jupyter Notebook, Google Colab

Installed packages: `transformers`, `torch`, `numpy`

Internet access to download Hugging Face models

Methodical instructions for laboratory work

Safety Requirements

Work in a ventilated room with a standardized level of lighting

Follow the rules of working at the computer, including breaks every 45-60 minutes

Use official repositories and libraries

Avoid passing sensitive data to the model

Save files in a timely manner and check the work of the code on small examples

Typical tasks

1. Use the `distilbert-base-uncased` model for the classification task: to recognize the positive or negative tone of the text
2. Use the `bert-base-cased` model to extract named entities from professional text
3. Using the `question-answering` pipeline, implement a system for answering a question for a given paragraph

Control questions

1. What tasks can be solved using the `transformers` library
2. What is a pre-trained model and why is it used?
3. What is the role of the tokenizer in the model?
4. What functions does the `pipeline` object provide?
5. How to download a model from the Hugging Face repository
6. What is a task-specific pipeline and give examples
7. What is the input and output structure for classification models
8. Which models are used to extract entities
9. How to control response generation parameters in QA tasks
10. In what applied tasks can transformers be used in your professional field?

PRACTICAL WORK №13

Topic: Running Large Language Models with Ollama

Purpose of the work

Explore the possibilities of running and interacting with large language models (LLMs) in an on-premises environment using the Ollama tool. Gain hands-on experience generating responses and handling user requests using models such as LLaMA, Mistral, Gemma, and more.

Theoretical substantiation

Large Language Models (LLMs) have become the basis of modern artificial intelligence systems capable of generating texts, conducting dialogue, translating, answering questions, summarizing and many other functions. The development of such models requires significant computing resources, but the launch of their local versions is made possible by tools like **Ollama**.

Ollama is a framework that allows you to run and use an LLM on your local computer in a simple and convenient way. It provides access to many popular models, supports command line and API communication, and is compatible with Jupyter and other Python frameworks.

In professional activities, the local launch of an LLM makes it possible to ensure privacy, autonomy and customization of models to the needs of specific tasks - from automatic reporting to text mining.

Equipment and materials

A personal computer with Windows, macOS, or Linux

At least 8 GB of RAM (preferably from 16 GB), a discrete GPU is welcome

The Ollama tool is installed (<https://ollama.com/>)

Internet access for the first download of models

Python environment (e.g. Jupyter Notebook) if API integration is required

Methodical instructions for laboratory work

Safety Requirements

Work in standardized conditions, observing the requirements for the duration and organization of work

Monitor disk space usage and CPU/GPU resources

Not to use Ollama models to process personal or sensitive data

When launching and interacting with the LLM, comply with the rules of responsible use

Work only with officially supported models and builds

Typical tasks

1. Install Ollama on your local machine and download one of the models: `llama2`, `mistral`, `gemma`
2. Conduct a dialogue with the model through the terminal, setting at least 3 thematic requests of a professional orientation
3. Set up a simple application (CLI or script) that receives input text and returns a response from the Ollama model via the API

Control questions

1. What is Ollama and what are its advantages compared to cloud-based LLM services?
2. Which language models can be run through Ollama
3. What are the minimum system requirements to work with Ollama
4. What is the difference between running an LLM locally and working through APIs (e.g., OpenAI, YandexGPT)
5. How to interact with the Ollama model through the terminal
6. What is prompt and how to formulate it correctly
7. How you can integrate Ollama with Python code
8. What limitations do local models have?
9. What security aspects to consider when working with an LLM
10. What are the potential applications of Ollama in your professional field

PRACTICAL WORK №14

Topic: Working with YandexGPT

Purpose of the work

Familiarization with the possibilities of using the YandexGPT model to generate texts, answer questions and support professional tasks. Mastering interaction with the model through a web interface and API.

Theoretical substantiation

YandexGPT is a large language model developed by Yandex designed for natural language processing in Russian. It is able to generate texts, summarize documents, answer questions, conduct a dialogue, and perform intellectual user support tasks.

Unlike models hosted on foreign platforms, YandexGPT provides the advantages of localization, compliance with Russian security requirements, as well as higher quality generation for the Russian language. The model is available through the Yandex Cloud web interface and an integrated API, which allows it to be used in professional solutions: chatbots, support services, and expert systems.

In a professional environment, YandexGPT can be used for automatic text generation, data analysis, generation of answers in dialogue systems, intelligent search, and as a decision support tool.

Equipment and materials

Personal computer with Internet access

Yandex account (to access Yandex Cloud)

Browser or Python environment with libraries `requests`, `json`

If necessary: access to the Yandex Cloud API and authorization token

Methodical instructions for laboratory work

Safety Requirements

Do not use personal or confidential data in requests

Work only in an authorized and controlled environment

Follow the rules for secure work with external APIs

Save intermediate and final results of work

Work in conditions that meet sanitary and ergonomic requirements

Typical tasks

1. Get access to the YandexGPT model through the web interface or Yandex Cloud API
2. Formulate 3-5 professionally oriented queries (search for information, resume, generation of instructions)
3. Write a Python script that interacts with YandexGPT via API and processes input text

Control questions

1. What is YandexGPT and how does it differ from other LLMs?
2. What tasks can be solved using this model
3. How to access the YandexGPT API
4. What is an authorization token and how is it used?
5. How to Formulate Effective Prompt Queries
6. What restrictions apply when working with YandexGPT
7. What are the ways to integrate the model into applications?
8. How to control the length and format of the model response
9. What areas of professional activity benefit from the implementation of YandexGPT
10. How YandexGPT can be useful for automating text analytics

PRACTICAL WORK №15

Topic: Reinforcement learning: implementing tabular Q-Learning on the example of the GridWorld task

Purpose of the work

Mastering the basic principles of reinforcement learning by implementing the Q-Learning algorithm in a discrete environment. Developing skills in building and updating the Q table, choosing actions, and analyzing the agent's strategy.

Theoretical substantiation

Reinforcement Learning (RL) is an approach in machine learning in which an agent learns by interacting with the environment and receiving a reward for actions. Unlike supervised learning, there is no pre-labeled data, and the target function is formed based on experience.

The Q-Learning algorithm refers to value-based learning methods. It is aimed at teaching the optimal strategy for choosing actions with the maximization of the total reward. For each "state-action" pair $Q(s,a)$, a utility score is stored. The update is made according to the Bellman formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)]$$

where:

- s is the current state,
- a is the selected action,
- r is the reward,
- s' is the next state,
- α is the learning rate,
- γ is the discount factor.

An example of an environment for such a task would be GridWorld, a two-dimensional field through which an agent navigates to a goal while avoiding obstacles. This is a simple but illustrative model that illustrates learning through experience. The implementation of Q-Learning in GridWorld allows you to demonstrate how an agent adapts behavior and forms a strategy.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Development environment: Jupyter Notebook, Google Colab

Python Interpreter 3.7+

Installed packages: `numpy`, `matplotlib`, (optionally — `Gymnasium`, `seaborn`)

Methodical instructions for laboratory work

Safety Requirements

Work should be done in a well-lit and organized workplace

Follow the rules of working at the computer: breaks every 45-60 minutes

Do not use third-party libraries and environments without permission

Monitor the load on the system when starting training cycles

Do not apply trained models outside of training tasks without checking

Typical tasks

1. Implement a 5×5 GridWorld environment where the target is in the bottom right corner and multiple cells are negative reward "traps"
2. Implement a tabular Q-Learning with the parameters: $\epsilon = 0.1$, $\alpha = 0.5$, $\gamma = 0.9$. Train an agent on 1000 episodes
3. Visualize the resulting Q-table and build the trajectory of the agent's movement from the starting point while following the optimal strategy

Control questions

1. What is reinforcement learning?
2. What is the purpose of using the Q table in the algorithm?
3. What do α , γ , and ϵ parameters mean in Q-Learning?
4. Why ϵ -greedy strategy is used?
5. What is the difference between supervised learning and reinforcement learning?
6. Why the Q-Learning algorithm can be considered off-policy?
7. What are the limitations of tabular Q-Learning?
8. What happens if α learning factor is too high or low?

9. How to interpret values in a Q-table
10. Where Q-Learning can be used in the professional field

PRACTICAL WORK No16

Topic: Comparison of Active and Passive Reinforcement Learning

Purpose of the work

To study the differences between active and passive reinforcement learning, to master the methods of evaluating policy and building an optimal strategy using the example of a discrete environment. To learn how to analyze the effectiveness of agent behavior under fixed and teachable policies.

Theoretical substantiation

In reinforcement learning, the agent studies the interaction with the environment, seeking to maximize the total reward. There are two approaches:

- **Passive learning** – the agent examines behavior in the environment under a given (fixed) policy and seeks to assess its quality without changing the strategy.
- **Active learning** – the agent simultaneously assesses the environment and changes the strategy based on the accumulated experience, learning to choose the best actions.

The passive approach is useful in cases where the policy is set from the outside (for example, by experts) and its evaluation is required. The active approach is implemented in classical algorithms such as Q-Learning or SARSA, where the strategy is formed by the agent based on experience and feedback.

This paper uses **the FrozenLake** environment from the `gymnasium` package, which simulates the task of moving on a slippery surface in order to achieve a goal while avoiding pits. This is a stochastic environment, which makes it suitable for strategy analysis.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Development environment: Jupyter Notebook, Google Colab

Python programming language version 3.8+

Installed packages: `Gymnasium`, `numpy`, `matplotlib`

Methodical instructions for laboratory work

Safety Requirements

Comply with the standards of work organization and workplace lighting

Use only teacher-APPROVED BYd libraries

Regularly save intermediate results and check the correctness of the code

Avoid excessive compute overhead

Do not use external or unofficial implementations of the environment

Typical tasks

1. Start the `FrozenLake-v1` environment and implement passive evaluation of the specified policy using **the Policy Evaluation algorithm**
2. Implement the active Q-Learning algorithm and train the agent in the same environment
3. Compare the behavior and the final strategy in both cases in terms of the number of successful episodes and the average reward

Control questions

1. What is the fundamental difference between active and passive reinforcement learning?
2. What an agent policy means and how it is formed
3. When is it advisable to use passive learning?
4. What tasks can be solved with the help of active learning
5. How the Value Function is Evaluated
6. What is an Action-Value Function?
7. Why is the FrozenLake environment considered stochastic
8. What metrics can be used to compare strategies

9. How ϵ -greedy strategy affects active learning
10. What are the limitations of the passive approach?

PRACTICAL WORK №17

Topic: Basics of image processing using OpenCV

Purpose of the work

Learn basic image processing operations using the OpenCV library. Master the methods of reading, displaying, transforming, and filtering images as a stage of data preparation in computer vision systems.

Theoretical substantiation

Image processing is a fundamental step in computer vision systems, the quality of which determines the success of object recognition, classification, tracking, and analysis. Modern approaches involve the use of specialized libraries, such as **OpenCV** (Open Source Computer Vision Library), which provides a wide range of tools for capturing, processing, and analyzing images.

The main operations include:

- **Uploading and displaying images** in various color formats (BGR, RGB, Grayscale);
- **Changing the size and orientation of images** (scaling, rotating);
- **Filtering** (blur, median filter, Gaussian filter);
- **Boundary detection** with Sobel, Laplace, and Canny operators;
- **Work with colors** (translation between color spaces, masking by color).

Mastering these methods allows you to prepare images for subsequent analysis, which is especially important in professional tasks of automation of visual control, diagnostics, video surveillance, document analysis, etc.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Development environment: Jupyter Notebook, Google Colab

Python interpreter version 3.8 and above

Installed packages: `opencv-python`, `numpy`, `matplotlib`

A set of images in JPG/PNG format (prepared by the instructor or downloaded from open sources)

Methodical instructions for laboratory work

Safety Requirements

Work must be carried out in an ergonomic environment with the correct organization of the workplace

Observe the time limit at the computer (breaks every 45-60 minutes)

Avoid processing personal and confidential images

Save intermediate processing results and check the correctness of the code

Follow the instructor's instructions when uploading external images

Typical tasks

1. Read and display the image in color and grayscale, save the result
2. Resize the image, rotate it 90 degrees, and flip it horizontally
3. Apply three filters to the image: Gaussian blur, median filter, and Canny operator to highlight borders

Control questions

1. What image formats does OpenCV support?
2. What is the difference between BGR and RGB color spaces?
3. How to resize an image and why you need it
4. What is filtering and what filters are used in OpenCV
5. What is the Canny operator used for?

6. How to Display Multiple Images on a Single Chart
7. What types of image distortion can interfere with its analysis
8. What is the difference between a median filter and a Gaussian filter
9. What are the tasks in which image processing is used in the professional field?
10. What limitations does OpenCV have when working with large images?

PRACTICAL WORK No18

Topic: Detection and classification of objects in images using Haar cascades and neural networks

Purpose of the work

Master methods for detecting objects in images using the Haar cascade algorithm and pre-trained neural network models. Gain practical skills in using OpenCV tools and neural networks for object recognition tasks.

Theoretical substantiation

Object detection is one of the key tasks of computer vision, which is to localize and identify objects in an image. There are two main approaches:

- **Haar cascades** are a classic method based on cascading classifiers and Haar features. It is effective for detecting simple objects (for example, faces) in conditions of limited computing resources.
- **Neural networks** are modern techniques based on deep convolutional architectures (e.g., MobileNet, YOLO, SSD) that allow both localization and classification of objects with high accuracy.

The OpenCV library provides a user-friendly interface for working with Haar cascades and integrating with neural network models downloaded in ONNX format or through the `dnn` module. These approaches are widely used in video surveillance, image mining, robotics, medicine, and industry.

Equipment and materials

A personal computer with Windows, macOS, or Linux

Development environment: Jupyter Notebook, Google Colab

Python programming language version 3.8 and higher

Installed packages: `opencv-python`, `numpy`, `matplotlib`

Haar cascade files (`haarcascade_frontalface_default.xml`)

Pre-trained neural network models (e.g., MobileNet SSD or YOLO in ONNX format, or `.pb`)

Images with objects (faces, cars, signs, etc.)

Methodical instructions for laboratory work

Safety Requirements

Work in an organized, ventilated room

Avoid overloading computing resources when processing video and large images

Do not use confidential or personal images

Comply with the rules for handling files and network resources

Control the correctness of saving and displaying results

Typical tasks

1. Use the built-in OpenCV classifier to detect faces in an image using Haar cascades
2. Load a pre-trained neural network model and detect objects in the photo using the `cv2.dnn` module
3. Compare the quality and speed of the Haar cascade and the neural network on the same set of images

Control questions

1. What is a Haar cascade and how does it work?
2. What are the advantages and disadvantages of Haar cascades
3. What is the difference between object detection and classification?

4. How the neural network model is loaded and used in OpenCV
5. What is input image preprocessing for neural networks?
6. How to Implement Bounding Boxes
7. What are the input image requirements in `cv2.dnn`
8. What parameters affect the accuracy and performance of detection
9. How to Interpret Confidence Score in Neural Network Results
10. Where object detection methods are used in professional activities

CONCLUSION

Practical works presented within the framework of the discipline "*Artificial Intelligence in the Professional Field*" are an integral part of the educational process and are aimed at forming students' practical skills in applying artificial intelligence methods to real tasks of professional activity.

In the course of laboratory tasks, students master key algorithms and technologies: from uninformed and heuristic search strategies to local methods of optimization and reinforcement learning, from basic image processing to working with modern language models and neural network detection tools. Particular attention is paid to the implementation of algorithms in Python programming environments, the use of specialized libraries (NetworkX, OpenCV, transformers, gymnasium, Ollama, YandexGPT) and the interpretation of the results in an applied context.

A set of laboratory works contributes to the development of systems thinking, algorithmic culture, the ability to engineering modeling and critical analysis. The orientation of tasks to professionally-oriented cases strengthens the interdisciplinary focus of training specialists and creates the basis for the further development of advanced AI tools in research and applied activities.

The performance of laboratory work in the discipline provides a practice-oriented assimilation of theoretical provisions, contributes to the development of the competence of UK-1 and forms readiness to use intelligent systems in their professional field.

EDUCATIONAL, METHODOLOGICAL AND INFORMATION SUPPORT OF THE DISCIPLINE

Reference citations:

Bolotsky, A. V.; Investigation of operations and optimization methods Electronic resource / Bolotsky A. V., Kochetkova O. A. - St. Petersburg : Lan, 2020. - 116 p. - ISBN 978-5-8114-4568-4

Saraev, P. V.; Methods of Machine Learning Electronic resource: Guidelines and tasks for laboratory work on the course / P. V. Saraev. - Lipetsk : Lipetsk State Technical University, EBS ASV, 2017. - 48 p. - The book is in the premium version of EBS IPR BOOKS. - ISBN 2227-8397

Sopov, E. A.; Multi-Criteria Neuroevolutionary Systems in Machine Learning and Human-Machine Interaction Electronic resource / Sopov E. A., Ivanov I. A. : monograph. - Krasnoyarsk : SibFU, 2019. - 160 p. - ISBN 978-5-7638-3969-2

Further reading:

Methodical recommendations for practical classes in the discipline "Theory and Practice of Machine Translation": Direction of training 45.04.02 Linguistics. Master's program "Translation and Translation Studies". Qualification of the graduate - master. Form of study - full-time. Curriculum of 2015. Studied in the 3rd semester. - Stavropol : NCFU, 2015. - 9 p.

Nedelko, V. M.; Fundamentals of Statistical Methods of Machine Learning Electronic resource: Textbook / V. M. Nedelko. - Novosibirsk: Novosibirsk State Technical University, 2010. - 72 p. - The book is in the premium version of EBS IPR BOOKS. - ISBN 978-5-7782-1385-2

Rapakov, G. G.; Methods and algorithms of machine learning in making management

decisions in the regional system of medical prevention (experience of the Vologda region)
Elektronnyi resurs / Rapakov G. G., Kasimov R. A. : monograph. - Vologda : VoSU, 2014. - 143
p. - ISBN 978-5-87851-548-1

Educational and methodical literature:

Saraev, P. V.; Methods of Machine Learning Electronic resource: Guidelines and tasks for
laboratory work on the course / P. V. Saraev. - Lipetsk : Lipetsk State Technical University, EBS
ASV, 2017. - 48 p. - The book is in the premium version of EBS IPR BOOKS. - ISBN 2227-8397