Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт экономики и менеджмента

УТВЕРЖДАЮ:
Директор Института
экономики и менеджмента
_____ Е.В. Нехода
« 20 » 04 2023 г.

Рабочая программа дисциплины

Искусственный интеллект в экономике

по направлению подготовки

38.04.01 Экономика

Направленность (профиль) подготовки:
«Экономика»

Форма обучения
Очная

Квалификация
Магистр

Год приема
2023

СОГЛАСОВАНО:
Руководитель ОП
_____ Н.А. Скрыльникова

Председатель УМК
_____ М.В. Герман

Томск – 2023

1. Цель и планируемые результаты освоения дисциплины (модуля)

Целью освоения дисциплины является формирование следующих компетенций:

– ПК-2 – Способен разрабатывать стратегии управления изменениями в организации;

Результатами освоения дисциплины являются следующие индикаторы достижения компетенций:

ПК-2.4. Определяет основные аспекты организации, которые могут быть затронуты стратегическими изменениями;

2. Задачи освоения дисциплины

Целью освоения дисциплины «Искусственный интеллект в экономике» является формирование основ теоретических знаний и практических навыков работы в области основных стратегий искусственного интеллекта: экспертных систем и искусственных нейронных сетей. В рамках дисциплины рассматриваются теоретические основы построения искусственных нейронных сетей, а также практические вопросы использования нейросетевых технологий для решения экономических задач.

Задачи дисциплины:

- знакомство слушателей с методами искусственного интеллекта (ИИ);
- сформировать у обучающихся навыки использования методов и алгоритмов теории ИИ;
- дать представление о возможностях аппарата теории ИИ и способах анализа сложных задач;
- ознакомить студентов с современной методологической базой нейросетевых технологий;
- показать эффективность использования нейросетевых технологий для решения экономических задач, очертить круг задач, решаемых методами нейросетевого моделирования.

3. Место дисциплины (модуля) в структуре образовательной программы

Дисциплина относится к части образовательной программы, формируемой участниками образовательных отношений, предлагается обучающимся на выбор.

4. Семестр(ы) освоения и форма(ы) промежуточной аттестации по дисциплине

Семестр 3, зачет.

5. Входные требования для освоения дисциплины

Для успешного освоения дисциплины требуются результаты обучения по следующим дисциплинам: «Эконометрика», «Pyton и R для анализа данных», «Анализ и прогнозирование временных рядов», «Большие данные», «Интеллектуальный анализ данных»

6. Язык реализации

Русский

7. Объем дисциплины (модуля)

Общая трудоемкость дисциплины составляет 3 з.е., 108 часов, из которых:

– лекции: 8 ч.;

– практические занятия: 20 ч.;

Объем самостоятельной работы студента определен учебным планом.

8. Содержание дисциплины (модуля), структурированное по темам

Тема 1. Введение в искусственный интеллект
Введение в искусственный интеллект.

Тема 2. История и развитие ИИ по областям
История развития искусственного интеллекта в разных областях. Информационное направление ИИ. Применение и становление искусственного интеллекта. Влияние искусственного интеллекта на прикладные исследования и экономику.

Тема 3. ИИ в экономике
Использование искусственного интеллекта в экономике и прикладных областях. Методы и алгоритмы решения экономических задач.

Тема 4. Экспертные системы
Понятие экспертной системы. Виды экспертных систем. Применение экспертных систем. Классификация и применяемые алгоритмы в экспертных системах.

Тема 5. Искусственные нейронные сети
Понятие искусственной нейронной сети (ИНС). Персептрон и его применение. Виды ИНС. Методы обучения ИНС. Задачи систем искусственного интеллекта. Решение задач с помощью ИНС и визуализация данных.

Тема 6. Генетические алгоритмы
Понятие и виды генетических алгоритмов. Особенности решения задач с помощью генетических алгоритмов. Принципы использования и применения.

Тема 7. Решение экономических задач методами ИИ
Общие способы решения задач. Основные виды логических выводов.

Практическая работа №1. Классификатор на основе перцептрона
Цель: разработать нейронную сеть для классификации данных на Python.

1. Установите с официального сайта или репозитория Python на целевую операционную систему (https://www.python.org/downloads/windows/)
2. Установите библиотеку NeuroLab для Python (https://pythonhosted.org/neurolab/install.html)
3. Установите библиотеку numpy для Python (Пример: >py -m pip install numpy)
4. Установите библиотеку matplotlib для Python
5. Установите библиотеку perceptron для Python (https://pypi.org/project/perceptron/)
6. Создайте новый файл Python и импортируйте файлы и библиотеки (numpy, matplotlib, neurolab)
    1. import numpy as np
    2. import matplotlib.pyplot as plt
    3. import neurolab as nl
7. Скачайте и загрузите входные данные из текстового файла data_lab1.txt.
    1. text = np.loadtxt('C:\py\data_lab1.txt')
8. Разделите текст на точки данных и метки

1. data = text[:, :2]
2. labels = text[:, 2] .reshape((text.shape[0], 1))

9. Постройте график точек данных
    1. plt. figure()plt.scatter(data[:,0], data[:,1])
    2. plt.xlabel ('Размерность 1')
    3. plt.ylabel ('Размерность 2')
    4. plt.title ('Входные данные')
10. Определите максимальное и минимальное значения, которые могут достигаться в каждом измерении.

    1. diml_min, diml_max, dim2_min, dim2_max = 0, 1, 0, 1
11. Задайте количество нейронов в выходном слое
    1. num_output = labels.shape[1]
12. Определите перцептрон с двумя входными нейронами
    1. diml = [diml_min, diml_max]
    2. dim2 = [dim2_min, dim2_max]
    3. perceptron = nl.net.newp([diml, dim2], num_output)
13. Обучите перцептрон с помощью тренировочных данных
    1. error_progress = perceptron.train(data, labels, epochs=100, show=20, lr=0.03)
14. Отобразите график процесса обучения, используя метрику ошибки
    1. plt.figure()
    2. plt.plot(error_progress)
    3. plt.xlabel('Количество эпох')
    4. plt.ylabel('Ошибка обучения')
    5. plt.title('Изменение ошибки обучения')
    6. plt.grid()
    7. plt.show()
15. Оформите отчет с ходом ваших действий и результатами.

Практическая работа №2. Многослойной нейронная сеть

Цель: построить многослойную нейронную сеть с высокой точностью

Задание: Создайте нейронную сеть, которая будет иметь более одного слоя для извлечения базовых закономерностей, существующих среди тестовых данных.

1. Создайте новый файл Python и импортируйте файлы и библиотеки (numpy, matplotlib, neurolab).
2. Сгенерируйте выборочный набор точек данных, используя уравнение $y = 2x + x^2 + 3$, а затем нормализуйте данные.
3. Сгенерируйте тренировочные данные
    1. min_val = -20
    2. max_val = 20

3. num_points = 150
4. x = np.linspace(min_val, max_val, num_points)
5. y= 2*x + np.square(x) + 3
6. y /= np.linalg.norm(y)

4. Переформируйте приведенные выше переменные для создания тренировочного набора данных.
    1. data = x.reshape(num_points, 1)
    2. labels = y.reshape(num_points, 1)
5. Постройте график входных данных.
    1. plt.figure()
    2. plt.scatter(data, labels)
    3. plt.xlabel('Размерность 1')
    4. plt.ylabel('Размерность 2')
    5. plt.title('Входные данные')
6. Определите многослойную нейронную сеть с двумя скрытыми слоями. Создайте 20 нейронов в первом слое и 12 нейронов во втором слое. Задача заключается в предсказании одного значения, поэтому выходной слой будет содержать всего один нейрон.
    1. nn = nl.net.newff([[min_val, max_val]], [20, 12, 1])
7. Установите метод градиентного спуска в качестве обучающего алгоритма.
    1. nn.trainf = nl.train.train_gd
8. Обучите нейронную сеть, используя сгенерированный ранее тренировочный набор данных.
    1. error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)
9. Запустите нейронную сеть для тренировочных точек данных.
    1. output = nn.sim(data)
    2. y_pred = output.reshape(num_points)
10. Постройте график продвижения процесса обучения.
    1. plt.figure()
    2. plt.plot(error_progress)
    3. plt.xlabel('Количество эпох')
    4. plt.ylabel('Ошибка обучения')
    5. plt.title('Изменение ошибки обучения')
11. Постройте график предсказанных результатов.
    1. x_dense = np.linspace(min_val, max_val, num_points * 2)
    2. y_dense_pred = nn.sim(x_dense.reshape(x_dense.size,1)).reshape(x_dense.size)
    3. plt.figure()
    4. plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
    5. plt.title('Фактические и прогнозные значения')
    6. plt.show()

Практическая работа №3. Построение векторного квантизатора

Цель работы: построить векторный квантизатор с использованием искусственных нейронных сетей

Справка: Векторная квантизация – это метод квантизации, в котором входные данные представляются фиксированным количеством представительных точек. Этот подход широко применяется при распознавание изображений, семантический анализ и наука о данных.

Задание: построить векторный квантизатор.

1. Измените файл конфигурации библиотеки neurolab для работы с вещественными числами.
    1. Откройте файл net.py любым редактором для правки (путь к файлу конфигурации \AppData\Local\Programs\Python\Python311\Lib\site-packages\neurolab)
    2. Найдите 176 строчку inx = np.floor(cn0 * pc.cumsum()) и замените на inx = np.floor(cn0 * pc.cumsum()).astype (int)
2. Создайте новый файл Python и импортируйте файлы и библиотеки (numpy, matplotlib, neurolab).
3. Импортируйте входные данные из файла data_lab3.txt. Каждая строка этого файла содержит шесть чисел. Первые два числа образуют
   точку данных, тогда как последние четыре - прямое кодирование метки.
   Всего имеются четыре класса данных
4. Импортируйте файл data_lab3.txt по аналогии с первой работой.
5. Разделите текст на данные и метки.
    1. data = text[:, 0:2]
    2. labels = text[:, 2:]
6. Определите нейронную сеть с двумя слоями: двадцать нейронов во входном слое и восемь в выходном.
    1. num_input_neurons = 20
    2. num_output_neurons = 8
    3. weights = [1/num_output_neurons] * num_output_neurons
    4. nn = nl.net.newlvq(nl.tool.minmax(data), num_input_neurons, weights)
7. Обучите нейронную сеть, используя тренировочный набор данных.
    1. _ = nn.train(data, labels, epochs=500, goal=-1)
8. Создайте сетку точек для визуализации кластеров
    1. xx, yy = np.meshgrid(np.arange(0, 10, 0.2), np.arange(0, 10, 0.2))
    2. xx.shape = xx.size, 1
    3. yy.shape = yy.size, 1
    4. grid_xy = np.concatenate((xx, yy), axis=1)
9. Выполните вычисления над сеткой точек данных с помощью нейронной сети.
    1. grid_eval = nn.sim(grid_xy)

10. Извлеките четыре класса.
    1. class_1 = data[labels[:,0] == 1]
    2. class_2 = data[labels[:,1] == 1]
    3. class_3 = data[labels[:,2] == 1]
    4. class_4 = data[labels[:,3] == 1]
11. Извлеките сетки, соответствующие этим четырем классам.
    1. grid_1 = grid_xy[grid_eval[:,0] == 1]
    2. grid_2 = grid_xy[grid_eval[:,1] == 1]
    3. grid_3 = grid_xy[grid_eval[:,2] == 1]
    4. grid_4 = grid_xy[grid_eval[:,3] == 1]
12. Постройте график результирующих данных.
    1. plt.grid(True)
    2. plt.plot(class_1[:,0], class_1[:,1], 'ko', class_2[:,0], class_2[:,1], 'ko', class_3[:,0], class_3[:,1], 'ko',class_4[:,0], class_4[:,1], 'ko')
    3. plt.plot(grid_1[:,0], grid_1[:,1], 'c.', grid_2[:,0], grid_2[:,1], 'bx', grid_3[:,0], grid_3[:,1], 'c^', grid_4[:,0], grid_4[:,1], 'y+')
    4. plt.axis([0, 10, 0, 10])
    5. plt.xlabel('Размерность 1')
    6. plt.ylabel('Размерность 2')
    7. plt.title('Векторная квантизация')
    8. plt.show()

Практическая работа №4. Нейронная сеть с обучением

Цель: разработать нейронную сеть и обучить с подкреплением.

*Для выполнения задания необходима версия Python 3.9.7

1. Создайте проект Python в Visual Studio
2. Установите библиотеку gym для Python (https://www.gymlibrary.dev/) через управление пакетами среды. Инструкция по установке доступна по ссылке (https://github.com/openai/gym#installation)
3. Установите библиотеку pyglet==1.5.27 для Python через управление пакетами среды.
4. Установите библиотеку pygame для Python через управление пакетами среды.
5. Импортируйте в проект библиотеки (argparse, gym)
6. Определите функцию для анализа входных аргументов.
    1. def build_arg_parser():
       parser = argparse.ArgumentParser(description='Run an environment')
       parser.add_argument('--input-env', dest='input_env', required=True,
       choices=['cartpole', 'mountaincar', 'pendulum', 'taxi', 'lake'],
       help='Specify the name of the environment')
       return parser
7. Определите основную функцию и проанализируйте входные аргументы.

1. if __name__=='__main__':
   args = build_arg_parser().parse_args()
   input_env = args.input_env

8. Создайте отображение входных аргументов на имена окружений, определенных в пакете OpenAI Gym
   1. name_map = {'cartpole': 'CartPole-v3',
      'mountaincar': 'MountainCar-v0',
      'pendulum': 'Pendulum-v1',
      'taxi': 'Taxi-v3',
      'lake': 'FrozenLake-v1'}
9. Создайте окружение на основании входного аргумента и сбросим его состояние.
   1. env = gym.make(name_map[input_env])
      env.reset()
10. Итерируйте 2000 раз, предпринимая действие на каждом шаге.
    1. for _ in range(2000):
       #Визуализируем окружение
       env.render()
       #выполняем случайное действие
       env.step(env.action_space.sample())
11. В стандартном средстве запуска Python в режиме отладки укажите аргумент сценария --input-env cartpole
12. Запустите проект и опишите результат.
13. Запустите проект с аргументами сценария (mountaincar, lake). Опишите результат.
14. Оформите отчет.

Практическая работа №5. Линейная регрессия с глубоким обучением

Цель: создать линейную регрессионную модель с использованием нейронных сетей.

В рамках поставленной цели создадим линейную регрессионную модель на основе перцептронов с помощью библиотеки TensorFlow. Это популярный набор средств глубокого обучения, который широко применяется для построенияразличных прикладных систем.

1. Создайте проект Python в Visual Studio
2. Выполните обновление пакета pip через управление пакетами среды.
   1. pip --upgrade
3. Установите библиотеку tensorflow==2.10.1 для Python (https://www.tensorflow.org/api_docs/python/tf) через управление пакетами среды (https://www.tensorflow.org/install/pip).
4. Импортируйте в проект библиотеки (numpy, matplotlib, tensorflow)
   1. import numpy as np
   2. import matplotlib.pyplot as plt

3. import tensorflow as tf
4. import tensorflow.compat.v1 as tf
5. tf.disable_v2_behavior()

5. Сгенерируем определенные точки данных и посмотрим, насколько нам удается подогнать под них модель. Определим количесгво генерируемых данных.

    1. num_points = 1200

6. Определим параметры, которые будут применяться для генерации данных. Используйте модель прямой линии: y = mx + c
    1. data = []
    2. m = 0.2
    3. c = 0.5
    4. for i in range(num_points):
7. Сгенерируйте 'x'
    1. x = np.random.normal(0.0, 0.8)
8. Сгенерируйте шум, варьирующий данные
    1. noise = np.random.normal(0.0, 0.04)
9. Вычислите значение y с помощью уравнения.
    1. y = m*x + c + noise
    2. data.append([x, y])
10. Разделите данные на входные и выходные переменные.
    1. x_data = [d[0] for d in data]
    2. y_data = [d[1] for d in data]
11. Постройте график сгенерированных данных
    1. plt.plot(x_data, y_data, 'ro')
    2. plt.title('Input data')
    3. plt.show()
12. Сгенерируйте веса и смещения для перцептрона. Для весов мы используйте генератор случайных чисел с равномерным законом распределения, а смещения задайте равными нулю.
    1. W = tf.Variable(tf.random.uniform([1], -1.0, 1.0))
    2. b = tf.Variable(tf.zeros([1]))
13. Определим уравнения для 'y', используя переменные TensorFlow
    1. y = W * x_data + b
14. Определите функцию потерь, которую можно будет использовать в процессе обучения. Оптимизатор будет пытаться минимизировать ее значение.
    1. loss = tf.reduce_mean(tf.square(y - y_data))
15. Определите оптимизатор, использующий метод градиентного спуска, и передайте ему функцию потерь
    1. optimizer = tf.train.GradientDescentOptimizer(0.5)

2. train = optimizer.minimize(loss)

16. Инициализируйте все созданные переменные.
    1. init = tf.initialize_all_variables()
17. Запустите сеанс работы с TensorFlow с помощью инициализатора.
    1. sess = tf.Session()
    2. sess.run(init)
18. Запустите процесс обучения. Укажите 10 итераций.
    1. num_iterations = 10
    2. for step in range(num_iterations):
19. Запустите сеанс
    1. sess.run(train)
20. Выведите данные о продвижении процесса обучения. По мере увеличения количества выполненных итераций параметр потерь непрерывно снижается.
    1. print('\nITERATION', step+1)
    2. print('W =', sess.run(W)[0])
    3. print('b =', sess.run(b)[0])
    4. print('loss =', sess.run(loss))

    Постройте график сгенерированных данных и наложите на него предсказательную модель. В качестве модели используется прямая линия.

    5. plt.plot(x_data, y_data, 'ro')
21. Постройте график предсказанной выходной линии
    1. plt.plot(x_data, sess.run(W) * x_data + sess.run(b))
22. Задайте параметры графика
    1. plt.xlabel('Размерность 0')
    2. plt.ylabel('Размерность 1')
    3. plt.title('Интеграция ' + str(step+1) + ' of ' + str(num_iterations))
    4. plt.show()
23. Закрывайте появляющиеся окна графиков до тех пор, пока линия не приблизится к реальной модели. Закрывая окно итерации окно, вы сможете наблюдать за ходом процесса обучения.
24. Оформите отчет

9. Текущий контроль по дисциплине

Текущий контроль по дисциплине проводится путем контроля посещаемости, проведения контрольных работ и фиксируется в форме контрольной точки не менее одного раза в семестр.

10. Порядок проведения и критерии оценивания промежуточной аттестации

Зачет в третьем семестре проводится в письменной форме по билетам. Билет содержит два теоретических вопроса. Практическая работа Продолжительность зачета 1,5 часа.

Примерный перечень теоретических вопросов.

1. Понятие ИИ. Его роль в современной экономике.
2. Области применения ИИ. Классификация искусственного интеллекта.
3. Интеллектуальный интерфейс. понятие и особенности.
4. История развития ИИ по областям.
5. Направления исследований ИИ.
6. ИИ в экономике. Искусственные нейронные сети. Применение и развитие систем.
7. Методы решения задач в ИИ.
8. понятие логики в ИИ. Методы доказательства в логике.
9. Оценка уровня искусственною интеллекта. Тест Тьюринга и пути решения.
10. Понятие нейронной сети. Обучение нейронных сетей.
11. Виды нейронных сетей. Генетические алгоритмы.
12. Однослойные и многослойные нейронные сети. Преимущество и особенности.
13. Понятие регрессии. Создание регрессора.
14. Обучение нейронной сети без учителя. Особенности, преимущества и недостатки.
15. Понятие смешанных гауссовских моделей. Создание классификаторов.
16. Обучение нейронной сети с подкреплением. Особенности и преимущества подхода.
17. Экспертные системы в экономике. Виды и роль в современной экономике. Преимущества и особенности.
18. Сверточные нейронные сети. Архитектура, особенности.
19. ИИ в экономике. Методы визуализации полученных результатов.

В основе оценивания ответов на зачёте лежат принципы объективности, справедливости и всестороннего анализа уровня знаний студентов.

При выставлении *«зачтено»* оценивается: знание фактического материала, а также культура речи, глубина знания, аргументированность ответа, связь теории и практики, умение решить задачу. Обязательным критерием является выполнение всех

«Не зачтено» ставится студенту, имеющему существенные пробелы в знании основного материала по программе и допустившему принципиальные ошибки при ответе на вопросы билета.

11. Учебно-методическое обеспечение

а) Электронный учебный курс по дисциплине в электронном университете «Moodle» - https://moodle.tsu.ru/course/view.php?id=33472

б) Оценочные материалы текущего контроля и промежуточной аттестации по дисциплине.

в) План семинарских / практических занятий по дисциплине.

г) Методические указания по проведению практических работ.

12. Перечень учебной литературы и ресурсов сети Интернет

а) основная литература:

– Шумский С. Воспитание машин: новая история разума : Научно-популярная литература. - Москва : ООО "Альпина нон-фикшн", 2021. - 174 с.. URL: http://znanium.com/catalog/document?id=387316.

– Бенгфорт Б. Прикладной анализ текстовых данных на Python : машинное обучение и создание приложений обработки естественного языка / Бенджамин Бенгфорт, Ребекка Билбро и Тони Охеда. - Санкт-Петербург [и др.] : Питер, 2019. - 363, [1] с.: ил. - (Бестселлеры OReilly).

– Матвеев М. Г. Модели и методы искусственного интеллекта. Применение в экономике : [учебное пособие для студентов вузов по специальности «Прикладная информатика (по областям)" и другим специальностям] / М. Г. Матвеев, А. С. Свиридов, Н. А. Алейникова. - Москва : Финансы и статистика [и др.], 2014. - 446, [1] с.: ил., табл.

б) дополнительная литература:

– Николенко С. И. Глубокое обучение : погружение в мир нейронных сетей / С. Николенко, А. Кадурин, Е. Архангельская. - Санкт-Петербург [и др.] : Питер, 2019. - 476 с.: ил., табл. - (Серия "Библиотека программиста").

– Мюллер А. Введение в машинное обучение с помощью Python : руководство для специалистов по работе с данными / Андреас Мюллер, Сара Гвидо. - Москва [и др.] : Диалектика, 2019. - 472, [1] с.: ил.

– Корячко В. П. Интеллектуальные системы и нечеткая логика : учебник : [для магистров вузов, обучающихся по направлению подготовки 2.09.04.01 "Информатика и вычислительная техника" (квалификация "магистр")] / В. П. Корячко, М. А. Бакулева, В. И. Орешков. - Москва : Курс, 2020. - 346, [1] с.: ил., табл.

в) ресурсы сети Интернет:

– ЭБС «Лань» https://e.lanbook.com/.

– ЭБС «Консультант студента» https://www.studentlibrary.ru/.

– ЭБС «Юрайт» https://urait.ru/.

– ЭБС ZNANIUM.com https://znanium.com/.

– Бесплатный курс по Python для начинающих https://code-basics.com/ru/languages/python.

– Программирование на Python https://stepik.org/course/67/promo.

– Введение в Python https://ru.hexlet.io/courses/python_101

– Основы машинного обучения https://openedu.ru/course/hse/INTRML/

– Машинное обучение https://ru.coursera.org/browse/data-science/machine-learning.

13. Перечень информационных технологий

а) лицензионное и свободно распространяемое программное обеспечение:

Для проведения лекционных и практических занятий необходимо лицензионное обеспечение: Операционная система Windows 7-10 или Linux, офисный пакет Microsoft Office 2013 или OpenOffice.

Для проведения практических занятий необходимо лицензионное программное обеспечение: ОС Windows 7-10 или Linux, свободно-распространяемый программный продукт Python, Visual Studio 2019.

Браузер Google Chrome/Opera/Firefox для работы в электронном курсе Moodle.

б) информационные справочные системы:

– Электронный каталог Научной библиотеки ТГУ – http://chamo.lib.tsu.ru/search/query?locale=ru&theme=system

– Электронная библиотека (репозиторий) ТГУ – http://vital.lib.tsu.ru/vital/access/manager/Index

– ЭБС Лань – http://e.lanbook.com/

– ЭБС Консультант студента – http://www.studentlibrary.ru/

– Образовательная платформа Юрайт – https://urait.ru/

– ЭБС ZNANIUM.com – https://znanium.com/

– ЭБС IPRbooks – http://www.iprbookshop.ru/

14. Материально-техническое обеспечение

Аудитории для проведения занятий лекционного типа.

Аудитории для проведения практических занятий, индивидуальных и групповых консультаций, текущего контроля и промежуточной аттестации.

Помещения для самостоятельной работы, оснащенные компьютерной техникой и доступом к сети Интернет, в электронную информационно-образовательную среду и к информационным справочным системам.

15. Информация о разработчиках

Погуда Алексей Андреевич, доцент кафедры информационного обеспечения инновационной деятельности факультета инновационных технологий, кандидат технических наук.