МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ (ШКОЛА)

СОГЛАСОВАНО
Руководитель ОП
_____ Величко А.С.
(подпись) (ФИО)

УТВЕРЖДАЮ
И.о. директора департамента
Заболоцкий В.С.
(подпись) (ФИО)
«_28_» декабря 2021 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ
Нейронные сети
Направление подготовки 01.04.04 Прикладная математика
(Аналитические, социальные и экономические сети)
Форма подготовки: очная

курс 2 семестр 4
лекции 36 час.
практические занятия 0 час.
лабораторные работы 36 час.
в том числе с использованием МАО лек. 0 час. / пр. 0 час. / лаб. 36 час.
всего часов аудиторной нагрузки 72 час.
в том числе с использованием МАО 36 час.
самостоятельная работа 144 час.
в том числе на подготовку к экзамену 36 час.
контрольные работы (количество) 3
курсовой проект не предусмотрен
зачет не предусмотрен
экзамен 4 семестр

Рабочая программа составлена в соответствии с требованиями Федерального государственного образовательного стандарта по направлению подготовки 01.04.04 Прикладная математика, утвержденного приказом Минобрнауки России от 10.01.2018 г. № 15.

Рабочая программа обсуждена на заседании департамента математики, протокол № 6 от 28 декабря 2021 г.

И.о. директора департамента математики Заболоцкий В.С.

Составитель: канд. физ.-мат. наук, доцент А.С. Величко

Владивосток
2021

Оборотная сторона титульного листа РПД

I. Рабочая программа пересмотрена на заседании департамента:

Протокол от «_____» _____ 20___ г. № _____

Директор департамента _____ _____
(подпись) (И.О. Фамилия)

II. Рабочая программа пересмотрена на заседании департамента:

Протокол от «_____» _____ 20___ г. № _____

Директор департамента _____ _____
(подпись) (И.О. Фамилия)

III. Рабочая программа пересмотрена на заседании департамента:

Протокол от «_____» _____ 20___ г. № _____

Директор департамента _____ _____
(подпись) (И.О. Фамилия)

IV. Рабочая программа пересмотрена на заседании департамента:

Протокол от «_____» _____ 20___ г. № _____

Директор департамента _____ _____
(подпись) (И.О. Фамилия)

АННОТАЦИЯ

Дисциплина «Нейронные сети» предназначена для студентов направления подготовки 01.04.04 «Прикладная математика», магистерская программа «Аналитические, социальные и экономические сети».

Общая трудоемкость освоения дисциплины составляет 6 зачетных единиц (216 часов). Дисциплина реализуется на 2 курсе в 4-м семестре. Дисциплина входит в дисциплины по выбору части, формируемой участниками образовательных отношений блока «Дисциплины (модули)».

Особенности построения курса: лабораторные работы (72 часа), самостоятельная работа (90 часов), подготовка к экзамену (54 часа).

Содержание дисциплины охватывает следующий круг вопросов: проведение статистического анализа и обработки данных с помощью методов нейронных сетей и инструментальных средств.

Цель – ознакомить с методами обработки массивов экономических данных в соответствии с поставленной задачей, научить анализировать, оценивать, интерпретировать полученные результаты и обосновывать выводы; строить статистические модели исследуемых процессов, явлений и объектов, относящихся к области профессиональной деятельности, анализировать и интерпретировать полученные результаты; выполнять статистическую обработку данных с помощью инструментальных средств.

Задачи:

- развитие способности анализировать и интерпретировать статистические данные, выявлять их тенденции;
- развитие готовности строить на основе описания ситуаций статистические модели,
- развитие способности анализировать и содержательно интерпретировать полученные результаты;
- развитие готовности прогнозировать динамику процессов и явлений на основе статистических моделей;

- развитие способности применять математические модели и методы для анализа и решения конкретных проблем, предлагать способы их решения.

Для успешного изучения дисциплины «Нейронные сети» у обучающихся желательны следующие предварительные компетенции:

- способностью применять аппарат математического анализа, линейной алгебры, теории вероятности и математической статистики;
- способностью работать с электронной таблицей Excel и программировать на языке Си.

В результате изучения данной дисциплины у обучающихся формируются следующие универсальные, общепрофессиональные, профессиональные компетенции (элементы компетенций).

Код и наименование профессиональной компетенции	**Код ПС (при наличии ПС) или ссылка на иные основания**	**Код трудовой функции (при наличии ПС)**	**Индикаторы достижения компетенции**
Тип задач профессиональной деятельности: **технологический**			
ПК-4 Способен организовывать статистические исследования	08.022 Статистик	С/01.7-04.7	ПК-4.1. Анализирует статистические данные на основе математических моделей и методов прикладной математики ПК-4.2 Использует программные средства для научной деятельности в статистике
Тип задач профессиональной деятельности: **научно-исследовательский**			
ПК-5 Способен к разработке и исследованию математических методов и моделей для проведения многовариантных аналитических расчетов и подготовки принятия решений	Анализ требований, предъявляемых к выпускникам	-	ПК-5.1. Формулирует модели, применяет методы анализа объектов, систем, процессов и технологий на основе математических моделей и методов прикладной математики ПК 5.2 Проводит сценарные аналитические расчеты для обоснования принимаемых решений по вариантам в том

			числе на основе программных средств

Код и наименование индикатора достижения компетенции	Наименование показателя оценивания (результата обучения по дисциплине)
ПК-4.1. Анализирует статистические данные на основе математических моделей и методов прикладной математики	Знает методы и модели нейронных сетей для анализа данных и соответствующих профессиональных стандартов
	Умеет организовывать работу по анализу статистических данных на основе методов нейронных сетей и имеет навыки по соответствующим профессиональным стандартам
	Владеет навыками проведения работ по анализу данных с использованием нейронных сетей и элементами трудовых функций соответствующих профессиональных стандартов
ПК-4.2 Использует программные средства для научной деятельности в статистике	Знает организацию научной деятельности в статистике на основе соответствующих профессиональных стандартов
	Умеет применять подходы и навыки научной деятельности в статистике по соответствующим профессиональным стандартам
	Владеет программными средствами при осуществлении научной деятельности в статистике и элементами трудовых функций соответствующих профессиональных стандартов
ПК-5.1. Формулирует модели, применяет методы анализа объектов, систем, процессов и технологий на основе математических моделей и методов прикладной математики	Знает математический аппарат, необходимый для решения задач с использованием моделей нейронных сетей
	Умеет применять соответствующую изучаемому процессу модели и методы нейронных сетей и проверять их адекватность
	Владеет навыками анализа результатов применения моделей нейронных сетей, принятия решений на основе полученных результатов
ПК 5.2 Проводит сценарные аналитические расчеты для обоснования принимаемых решений по вариантам в том числе на основе программных средств	Знает постановки задач нейронных сетей
	Умеет применять методы нейронных сетей для решения экономических и технических задач
	Владеет современным программным инструментарием нейронных сетей для анализа экономических и технических процессов и объектов

Для формирования вышеуказанных компетенций в рамках дисциплины «Нейронные сети» применяются неимитационные методы активного/интерактивного обучения: выполнение проектов с использованием компьютерных технологий и специализированного программного обеспечения.

I. СТРУКТУРА И СОДЕРЖАНИЕ ТЕОРЕТИЧЕСКОЙ ЧАСТИ КУРСА

не предусмотрена

II. СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСА

Лабораторные работы (84 час.)

Раздел I. Программный инструментарий для нейронных сетей.

Лабораторная работа №1. Начало работы с пакетом Python. Модули анализа данных. Графика (24 час.).

Лабораторная работа №2. «Глубокое обучение» и задачи классификации. (12 час.).

Раздел II. Свёрточные нейронные сети.

Лабораторная работа №3. Введение. Операция свёртки. Свёрточный и пулинг слои. Распознавание изображений. (12 час.)

Лабораторная работа №4. Техника Transfer Learning. Архитектуры. Анализ видео и аудио информации. (12 час.)

Раздел III. Рекуррентные нейронные сети.

Лабораторная работа №5. Введение. Языковые модели и генерация текста. (12 час.)

Лабораторная работа №6. Машинный перевод и распознавание речи. Sentiment analysis. (12 час.)

III. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Учебно-методическое обеспечение самостоятельной работы

обучающихся по дисциплине «Нейронные сети» представлено в Приложении 1 и включает в себя:

план-график выполнения самостоятельной работы по дисциплине, в том числе примерные нормы времени на выполнение по каждому заданию;

характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению;

требования к представлению и оформлению результатов самостоятельной работы;

критерии оценки выполнения самостоятельной работы.

IV. КОНТРОЛЬ ДОСТИЖЕНИЯ ЦЕЛЕЙ КУРСА

Контролируемые разделы дисциплины, этапы формирования компетенций, виды оценочных средств, зачетно-экзаменационные материалы, комплекты оценочных средств для текущей аттестации, описание показателей и критериев оценивания компетенций на различных этапах их формирования, методические материалы, определяющие процедуры оценивания знаний, умений и навыков и (или) опыта деятельности, а также критерии и показатели, необходимые для оценки знаний, умений, навыков и характеризующие этапы формирования компетенций в процессе освоения образовательной программы, представлены в Приложении 2.

V. СПИСОК УЧЕБНОЙ ЛИТЕРАТУРЫ И ИНФОРМАЦИОННО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература

1. Бруссард, М. Искусственный интеллект: пределы возможного / Мередит Бруссард ; пер. с англ. - Москва : Альпина нон-фикшн, 2020. - 362 с. - ISBN 978-5-00139-080-0. - Текст : электронный. - URL: https://znanium.com/catalog/product/1220958.

2. Келлехер, Д. Наука о данных: базовый курс / Джон Келлехер, Брендан Тирни ; пер. с англ.. - Москва : Альпина Паблишер, 2020. - 222 с. - ISBN 978-5-9614-3170-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1221800.

3. Чио, К. Машинное обучение и безопасность : руководство / К. Чио, Д. Фримэн ; перевод с английского А. В. Снастина. — Москва : ДМК Пресс, 2020. — 388 с. — ISBN 978-5-97060-713-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131707.

Дополнительная литература

1. Барский А.Б. Введение в нейронные сети : учебное пособие / Барский А.Б.. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 357 с. — ISBN 978-5-4497-0309-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/89426.html.

2. Джонс, М. Т. Программирование искусственного интеллекта в приложениях / М. Т. Джонс. — Москва : ДМК Пресс, 2011. — 312 с. — ISBN 978-5-94074-746-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/12441.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Ниворожкина Л. И., Арженовский С. Б. Многомерные статистические методы в экономике : Учебник. – М.: Дашков и К, 2008. .— 223 с. — Режим доступа: http://lib.dvfu.ru:8080/lib/item?id=chamo:264736&theme=FEFU

2. R — объектно-ориентированная статистическая среда. URL: http://herba.msu.ru/shipunov/software/r/r-ru.htm
3. А.Б. Шипунов, Е.М. Балдин, П.А. Волкова, А.И. Коробейников, С.А.Назарова, С.В. Петров, В.Г. Суфиянов. Наглядная статистика. Используем R! -- М.: ДМК Пресс, 2012. -- 298 с. ISBN 978-5-94074-785-828-1. URL: http://herba.msu.ru/shipunov/software/r/rbook-toc.htm
4. Статистическое программирование на R: Часть 1. Купаемся в изобилии статистических возможностей. URL: https://www.ibm.com/developerworks/ru/library/l-r1/
5. Необходимо ли вам изучать язык R? URL: http://www.ibm.com/developerworks/ru/library/bd-learnr/
6. Introduction to Applied Statistics: Lecture Notes. URL: https://people.richland.edu/james/lecture/m113/
7. Journal of Applied Statistics. URL: http://www.tandfonline.com/loi/cjas20#.VkrLPXbhDq4
8. Annals of Applied Statistics. URL: http://imstat.org/aoas/

Перечень дополнительных информационно-методических материалов

1. Дубров А.М., Мхитарян В.С., Трошин Л.И. «Многомерные статистические методы и основы эконометрики» - М., Московский государственный университет экономики, статистики и информатики. 2003. – 79 с.
2. Многомерный статистический анализ в экономических задачах. Компьютерное моделирование в SPSS : Учебное пособие / Под ред. И. В. Орловой. – М. : Вузовский учебник, 2009.
3. Маккинли У. Python и анализ данных [Электронный ресурс]. — М.: ДМК Пресс, 2015. — 482 с.
4. Саммерфилд М. Python на практике [Электронный ресурс]. — М.: ДМК Пресс, 2016. — 338 с.

5. Сузи Р.А. Язык программирования Python [Электронный ресурс]. М.: Интуит, 2007. — 328 с.
6. Справочник по прикладной статистике / Под ред. Э. Ллойда. - М.: Финансы и статистика, 1989.

Перечень информационных технологий и программного обеспечения

При осуществлении образовательного процесса по дисциплине используется свободно распространяемое программное обеспечение MS Excel, GNU R, Python.

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Рекомендации по планированию и организации времени, отведенного на изучение дисциплины, описание последовательности действий обучающихся

Освоение дисциплины следует начинать с изучения рабочей учебной программы, которая содержит основные требования к знаниям, умениям и навыкам. Обязательно следует учитывать рекомендации преподавателя, данные в ходе установочных занятий. Затем – приступать к изучению отдельных разделов и тем в порядке, предусмотренном программой.

Получив представление об основном содержании раздела, темы, необходимо изучить материал с помощью рекомендуемой основной литературы. Целесообразно составить краткий конспект или схему, отображающую смысл и связи основных понятий данного раздела и включенных в него тем. Обязательно следует записывать возникшие вопросы, на которые не удалось ответить самостоятельно.

Подготовку к началу обучения включает несколько необходимых пунктов:

1) Необходимо создать для себя рациональный и эмоционально достаточный уровень мотивации к последовательному и планомерному изучению дисциплины.

2) Необходимо изучить список рекомендованной основной и дополнительной литературы и убедиться в её наличии у себя дома или в библиотеке в бумажном или электронном виде.

3) Необходимо иметь «под рукой» специальные и универсальные словари, справочники и энциклопедии, для того, чтобы постоянно уточнять значения используемых терминов и понятий. Пользование словарями и справочниками необходимо сделать привычкой. Опыт показывает, что неудовлетворительное усвоение предмета зачастую коренится в неточном, смутном или неправильном понимании и употреблении понятийного аппарата учебной дисциплины.

4) Желательно в самом начале периода обучения возможно тщательнее спланировать время, отводимое на работу с источниками и литературой по дисциплине, представить этот план в наглядной форме (график работы с датами) и в дальнейшем его придерживаться, не допуская срывов графика индивидуальной работы и «аврала» в предсессионный период. Пренебрежение этим пунктом приводит к переутомлению и резкому снижению качества усвоения учебного материала.

Рекомендации по работе с литературой

1) Всю учебную литературу желательно изучать «под конспект». Чтение литературы, не сопровождаемое конспектированием, даже пусть самым кратким – бесполезная работа. Цель написания конспекта по дисциплине – сформировать навыки по поиску, отбору, анализу и формулированию учебного материала. Эти навыки обязательны для любого специалиста с высшим образованием независимо от выбранной специальности.

2) Написание конспекта должно быть творческим – нужно не переписывать текст из источников, но пытаться кратко излагать своими словами содержание ответа, при этом максимально структурируя конспект, используя символы и условные обозначения. Копирование и «заучивание» неосмысленного текста трудоемко и по большому счету не имеет большой познавательной и практической ценности.

3) При написании конспекта используется тетрадь, поля в которой обязательны. Страницы нумеруются, каждый новый вопрос начинается с нового листа, для каждого экзаменационного вопроса отводится 1-2 страницы конспекта. На полях размещается вся вспомогательная информация – ссылки, вопросы, условные обозначения и т.д.

4) В итоге данной работы «идеальным» является полный конспект по программе дисциплины, с выделенными определениями, узловыми пунктами, примерами, неясными моментами, проставленными на полях вопросами.

5) При работе над конспектом обязательно выявляются и отмечаются трудные для самостоятельного изучения вопросы, с которыми уместно обратиться к преподавателю при посещении установочных лекций и консультаций, либо в индивидуальном порядке.

6) При чтении учебной и научной литературы всегда следить за точным и полным пониманием значения терминов и содержания понятий, используемых в тексте. Всегда следует уточнять значения по словарям или энциклопедиям, при необходимости записывать.

7) При написании учебного конспекта обязательно указывать все прорабатываемые источники, автор, название, дата и место издания, с указанием использованных страниц.

Подготовка к промежуточной аттестации по дисциплине: экзамену (зачету)

К аттестации допускаются студенты, которые систематически в течение всего семестра посещали и работали на занятиях и показали уверенные знания в ходе выполнении практических заданий и лабораторных работ.

Непосредственная подготовка к аттестации осуществляется по вопросам, представленным в рабочей учебной программе. Тщательно изучите формулировку каждого вопроса, вникните в его суть, составьте план ответа. Обычно план включает в себя:

— определение сущности рассматриваемого вопроса, основных положений, утверждений, определение необходимости их доказательства;

— запись обозначений, формул, необходимых для полного раскрытия вопроса;

— графический материал (таблицы, рисунки, графики), необходимые для раскрытия сущности вопроса;

— роль и значение рассматриваемого материала для практической деятельности, примеры использования в практической деятельности.

VII. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Для осуществления образовательного процесса по дисциплине желательна учебная аудитория для проведения занятий лекционного типа и практических занятий: компьютерный класс.

Приложение 1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ (ШКОЛА)

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

по дисциплине «Нейронные сети»

Направление подготовки 01.04.04 Прикладная математика

магистерская программа «Аналитические, социальные и экономические сети»

Форма подготовки очная

Владивосток
2021

План-график выполнения самостоятельной работы по дисциплине

№ п/п	Дата/сроки выполнения	Вид самостоятельной работы	Примерные нормы времени на выполнение	Форма контроля
1	2 неделя	Повторение теоретического и практического материала дисциплины, заслушиваемого и конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины	23 часа	Собеседование
2	3 неделя	Самостоятельный разбор заданий и задач, решаемых на практических занятиях.	13 часов	Проект
3	4 неделя	Повторение теоретического и практического материала дисциплины, заслушиваемого и конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные проблемные вопросы по темам лекций	14 часов	Собеседование
4	5 неделя	Самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со специальным программным обеспечением	13 часов	Проект
5	6 неделя	Повторение теоретического и практического материала дисциплины, заслушиваемого и	14 часов	Собеседование

		конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные проблемные вопросы по темам лекций		
6	7 неделя	Самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со специальным программным обеспечением	13 часов	Проект
7	Сессия	Экзамен	54 часа	Экзамен

Характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению

Источник: Ниворожкина Л. И., Арженовский С. Б. Многомерные статистические методы в экономике : Учебник. – М.: Дашков и К, 2008. .— 223 с.

1. Темы 1-2, с. 36-42.
2. Тема 3, с. 61-71.
3. Тема 4, с. 119-129.
4. Тема 5, с. 90-96.
5. Тема 6, с. 141-151.
6. Тема 7, с. 168–173.

Источник: Дубров А.М., Мхитарян В.С., Трошин Л.И. «Многомерные статистические методы и основы эконометрики» - М., Московский государственный университет экономики, статистики и информатики. 2003. – 79 с.

7. Корреляционный анализ. Тренировочный пример. – с. 8-10
8. Регрессионный анализ. Тренировочный пример. – с. 18-21
9. Кластерный анализ. Тестовый пример. – с. 45-48
10. Компонентный анализ. Тренировочный пример. – с. 29-32
11. Компонентный анализ. Тренировочный пример. – с. 33-36

Источник: Маккинли У. Python и анализ данных. — М.: ДМК Пресс, 2015.

Интерпретатор Python

Python – *интерпретируемый* язык. Интерпретатор Python исполняет программу по одному предложению за раз. Стандартный интерактивный интерпретатор Python запускается из командной строки командой `python`:

```
$ python
Python 2.7.2 (default, Oct 4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 5
>>> print a
5
```

Строка `>>>` – это приглашение к вводу выражения. Для выхода из интерпретатора Python и возврата в командную строку нужно либо ввести команду `exit()`, либо нажать `Ctrl-D`.

Для выполнения Python-программы нужно просто набрать команду `python`, указав в качестве первого аргумента имя файла с расширением `.py`. Допустим, вы создали файл `hello_world.py` с таким содержимым:

```
print 'Hello world'
```

Чтобы выполнить его, достаточно ввести следующую команду:

```
$ python hello_world.py
Hello world
```

Многие программисты выполняют свой код на Python именно таким образом, но в мире *научных* приложений принято использовать IPython, улучшенный и дополненный интерпретатор Python. Ему посвящена вся глава 3. С помощью команды `%run` IPython исполняет код в указанном файле в том же процессе, что позволяет интерактивно изучать результаты по завершении выполнения.

```
$ ipython
Python 2.7.2 |EPD 7.1-2 (64-bit)| (default, Jul 3 2011, 15:17:51)
Type "copyright", "credits" or "license" for more information.
```

```
IPython 0.12 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: %run hello_world.py
```

```
Hello world

In [2]:
```

По умолчанию приглашение IPython содержит не стандартную строку `>>>`, а строку вида `In [2]:`, включающую порядковый номер предложения.

Источник: Онлайн-курс «Математика и Python для анализа данных». МФТИ, Академия Яндекса. https://www.coursera.org/learn/mathematics-and-python

Задание 1

1. Перейдите на сайт continuum.io по ссылке https://www.continuum.io/downloads
2. Выберите подходящую Вам операционную систему и перейдите в соответствующий раздел сайта.
3. Следуя инструкциям на сайте, установите Python 3.5.

Задание 2

1. Запустите ipython notebook. Команда запуска может несколько отличаться от команды запуска в инструкции из видео. Например, команда может быть такой: ipython-3.5 notebook
2. Создайте новый файл типа .ipynb
3. Создайте новую ячейку. В ней импортируйте библиотеку numpy (import numpy) и выведите на экран версию библиотеки (numpy.__version__)
4. Создайте новую ячейку. В ней импортируйте библиотеку scipy (import scipy) и выведите на экран версию библиотеки (scipy.__version__)
5. Создайте новую ячейку. В ней импортируйте библиотеку pandas (import pandas) и выведите на экран версию библиотеки (pandas.__version__)
6. Создайте новую ячейку. В ней импортируйте библиотеку matplotlib (import matplotlib) и выведите на экран версию библиотеки (matplotlib.__version__)
7. Сделайте скриншот №1, на котором будет хорошо видно результаты вашей работы, и загрузите его в форму.

Задание 3

1. Запустите ipython notebook. Команда запуска может несколько отличаться от команды запуска в инструкции, например, команда может быть такой ipython-2.7 notebook
2. Создайте новый файл типа .ipynb
3. В файле создайте новую ячейку и измените её тип на Markdown
4. В первой строке созданной ячейки наберите название специализации “Нейронные сети” и сделайте эту строку заголовком уровня 1.
5. В следующей строке созданной ячейки наберите название нашего курса “Математика и Python” и сделайте строку заголовком уровня 2.
6. В третьей строке ячейки наберите текст "Задание 1".
7. Запустите выполнение ячейки.
8. Сделайте скриншот №2, на котором будет хорошо видно результаты вашей работы, и загрузите его в форму.

В результате работы вы установите на компьютер Python и библиотеки, необходимые для дальнейшего прохождения курса.

Вы также научитесь запускать iPython notebook и выполнять в нем простые команды.

Выполнение задания будет проверяться на основе двух скриншотов, которые вы прикрепите к заданию.

Программирование на Python

1. Типы данных в Python

Настало время познакомиться с типами данных в Python. Числовые типы `int` и `float` уже были упомянуты ранее:

```
x = 3.5
print type(x) % OUT: <type 'float'>

y = 4
print type(y) % OUT: <type 'int'>
```

С помощью функции `type` можно узнать код произвольного объекта. Строки, например, могут быть двух типов:

```
print type("abs")  % OUT: <type 'str'>
print type(u"abs") % OUT: <type 'unicode'>
```

Отличия между ними будут подробнее обсуждаться позднее.

Упорядоченный набор значений в Python можно представить с помощью списка:

```
w = [1, 2, 3]
print type(w) % OUT: <type 'list'>
```

Списки в Python очень похожи на знакомые по другим языкам программирования массивы, но, в отличие от массивов, могут содержать элементы разных типов.

Python является языком с утиной типизацией. В языках с динамической типизацией, частным случаем которой является утиная типизация, тип переменной определяется не в момент её объявления, а в момент присваивания значения. Следовательно, в разных участках кода переменная может принимать значения разных типов. В случае утиной типизации границы применимости объекта определяются не столько конкретным типом и иерархией наследования, сколько наличием и отсутствия у него определенных методов и свойств. Термин происходит от шуточного «утиного теста»: «Если нечто выглядит, плавает и крякает как утка, то это, вероятно, и есть утка».

Другой способ представить упорядоченный набор значений состоит в определении кортежа (`tuple`). В Python кортежи записываются в круглых скобках:

```
print type((1, 2, 3)) % OUT: <type 'tuple'>
```

Кортеж относится к так называемым неизменяемым типам данных, чем существенно отличается от списка. Попытка изменить кортеж или добавить в него еще один элемент приведет к возникновению ошибки. Эта особенность позволяет использовать кортежи в качестве ключей словаря.

Неупорядоченный набор значений представим в виде множеств, которые записываются в фигурных скобках:

```
print type({1, 2, 3}) % OUT: <type 'set'>
```

Еще один тип данных, словарь, позволяет хранить в себе пары вида ключ-значение. Создать словарь можно с помощью конструктора `dict()`:

```
e = dict()      % Создан пустой словарь
e['abc'] = 3.5  % Строке-ключу 'abc' поставлено в
                %    соответствие значение 3.5 типа float
```

Ключами в словаре могут быть не только строки. В качестве ключа допустимо использовать, например, числовые значения и кортежи:

```
e[3.5]       = 'abc'
e[(1, 2, 3)] = 4.5
```


Но использование списка в качестве ключа недопустимо, так как список является изменяемым.

Синтаксис для чтения данных из словаря по ключу аналогичен синтаксису для получения данных по индексу списка. Грубо говоря, ключ есть обобщение понятия индекса списка: ключ может принимать значение произвольного хэшируемого типа. К слову, список не является хэшируемым, и именно поэтому не может быть выбран в качестве ключа.

В Python словарь использует хэш-таблицы. Идея заключается в том, чтобы заменить сравнение ключей более быстрым сравнением их хэш-значений. В таком случае важно, чтобы все ключи были хэшируемы, то есть хэш-значение не менялось во время исполнения программы, а равные ключи-объекты имели одинаковое хэш-значение.

Ранее было сказано, что множества являются изменяемыми объектами, а значит не могут выступать в роли ключей. Но в качестве ключей можно использовать неизменяемые множества, которые можно получить используя функцию `frozenset()`:

```
s = frozenset(1,2,3)
e[s] = 3.76
```

Таким образом, к данному моменту были изучены основные типы данных в Python и подробно было рассказано о существенных отличиях изменяемых и неизменяемых типов. На следующем занятии будут рассмотрены основные управляющие конструкции языка Python.

2. Синтаксис языка Python

В Python оператор условного перехода выглядит привычным образом. Например, пусть дан следующий код:

```
if x:
   print "OK"
else:
   print "NOT OK"
```

Если переменная `x` была равна `True`, то исполнится код в теле инструкции `if` и на экран будет выведено ÖK. В противном случае, например если x определена как `False`, исполнится код в теле инструкции `else`. В Python для выделения блоков кода используются отступы: отступы инструкций в пределах одного блока должны совпадать по величине.

Другой пример. Следующий код выводит числа от 0 до 9:

```
for i in range(10):
    print i
```

При этом в `range()` можно указать начальное значение i. Например, вот этот код выводит числа от 1 до 9.

```
for i in range(1,10):
```

```
    print i
```

В данных примерах каждое число выводится на отдельной строке, поскольку `print` по умолчанию начинает новую строчку после вывода требуемого значения. Чтобы все числа выводились в одной строке, необходимо добавить запятую в конец строчки с `print` следующим образом:

```
for i in range(1,10):
    print i,
```

Теперь числа от 1 до 9 выводятся в одной строке через пробел.

На самом деле `range()` это функция, которая возвращает список натуральных чисел в определяемом её аргументами диапазоне:

```
print range(2,5)
OUT: [2,3,4]
```

Использовать `range()` совершенно не обязательно. Циклы в Python перебирают по очереди элементы списка, стоящего после `in`. Например, можно написать так:

```
for i in [2,3,4]:
    print i,
```


В Python 2 кроме функции `range()` существует и функция `xrange()`, которую также можно использовать в цикле:

```
for i in xrange(2,5):
    print i,
OUT: 2, 3, 4
```

Использовать `range()` совершенно не обязательно. Циклы в Python перебирают по очереди элементы списка, стоящего после `in`. Например, можно написать так:

```
for i in [2,3,4]:
    print i,
```

2

В Python 2 кроме функции `range()` существует и функция `xrange()`, которую также можно использовать в цикле:

```
for i in xrange(2,5):
    print i,
OUT: 2, 3, 4
```

Результат выполнения не отличается от такового при использовании `range()`. Но можно убедиться, что `range` и `xrange` возвращают объекты разных типов:

```
print type(range(2,5))
OUT: list
print type(xrange(2,5))
OUT: iter
```

Функция `xrange` возвращает не список, а так называемый генератор. В то время, как при использовании `range()` сначала создается список, а только потом происходит итерирование по этому списку, при использовании генераторов список никогда не создается и не занимает память. Это особенно актуально, когда необходимо итерировать в очень большом диапазоне значений.

```
print xrange(2,500000000)
OUT:  xrange(2,500000000)
```

В Python существует способ удобно и компактно задавать некоторые списки с помощью так называемого конструктора списка (анг. list comprehension.):

```
w = [ x ** 2 for x in range(1,11) ]
print w
OUT: [ 1, 4, 9, 16, 25, 36, 49, 64, 81, 100 ]
print type(w)
OUT: <type 'list'>
```

В этом примере создан список квадратов чисел от 1 до 10. Если необходимо построить список, например, квадратов только четных чисел из этого диапазона, конструктор допускает использование условия:

```
w = [ x ** 2 for x in range(1,11) if x % 2 == 0]
OUT: [ 4, 16, 36, 64, 100 ]
```

Аналогично выглядит конструктор генераторов:

```
w = ( x ** 2 for x in range(1,11) if x % 2 == 0 )
OUT: ( 4, 16, 36, 64, 100 )
print type(w)
OUT: <type 'generator'>
```

В этом случае весь список не будет храниться в памяти во время работы цикла.

Также в Python доступен цикл `while`, который продолжает выполнение, пока выполнено указанное условие, а также операторы `break`, который досрочно прерывает цикл, и `continue`, который начинает следующий проход цикла, минуя оставшееся тело цикла. Например:

```
s = 0
while True: % Это условие всегда выполняется
    s += 1
    if s % 2 ==0: % Через раз
        print "Continue" % вместо s будет выведено "continue"
        continue % и код ниже в таком случае не будет исполнен в этой итерации
    print s
    if s > 10: % С помощью этой конструкции
        break  % <<бесконечный цикл досрочно прерывается>>
```

Важно уметь определять свои функции. Например, следующий код представляет собой альтернативную реализацию уже известной функции `range`:


```
def myrange(a,b):
    res = [ ]
    s = a
    while s!=b:
      res.append(s)
      s+=1
    return res
```

Этот пример демонстрирует синтаксис для создания функций. На самом деле при создании функции необходимо обработать особые случаи, но это не является текущей целью.

Функции в Python являются объектами первого класса: их можно передавать в качестве аргументов, присваивать их переменным и так далее. Например, функция `map()` позволяет обрабатывать одну или несколько последовательностей с помощью переданной в качестве аргумента функции:

```
def sq(x):
    return x ** 2

print map(sq, range(10))
OUT: [ 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Функция в Python может быть определена как с помощью оператора `def`, так и с помощью лямбда-выражения. Следующие операторы эквивалентны:

```
def sq(x):
    return x ** 2

sq = lambda x: x**2
```

Задание 4

Реализовать клиент-серверное приложение с использованием API GoogleMaps, позволяющее пользователю искать географические данные по введенному названию. Для этого воспользоваться сетевой библиотекой языка Python urllib.

Задания и методические рекомендации по теме «Компьютерная графическая визуализация данных в Mathplotlib»

1. Изучить рекомендуемую литературу по данной теме.
2. Изучить библиотеку mathplotlib для графической визуализации данных.
3. Применить методы из библиотеки mathplotlib для обработки и анализа данных.
4. Ознакомиться с другими средствами визуализации (см. ниже).

Источник: Маккинли У. Python и анализ данных. — М.: ДМК Пресс, 2015.

Инструментальная экосистема визуализации для Python

Как обычно бывает в проектах с открытым исходным кодом, в средствах создания графики для Python нехватки не ощущается (их слишком много, чтобы все перечислить). Помимо открытого кода, существуют также многочисленные коммерческие библиотеки с интерфейсом к Python.

В этой главе и в книге в целом я использую, в основном пакет matplotlib, потому что это наиболее популярное средство построения графиков в Python. Но хотя matplotlib является важной частью научной экосистемы Python, у него обнаруживается немало недостатков, когда дело доходит до создания и отображения статистических графиков. Пользователям MATLAB пакет matplotlib покажется знакомым, тогда как пользователи R (особенно работающие с великолепными пакетами `ggplot2` и `trellis`), наверное, испытают разочарование (по крайней мере, на момент написания этой книги). С помощью matplotlib можно создать отличные графики для отображения в веб, но для этого обычно приходится прикладывать немало усилий, потому что библиотека проектировалась для полиграфических изданий. Но если не обращать внимания на эстетические нюансы, то для большинства задач этого достаточно. В pandas я, наряду с другими разработчиками, стремился предложить пользователям удобный интерфейс, позволяющий упростить построение большинства графиков, применяемых в анализе данных.

Но существуют и активно используются и другие средства визуализации. Ниже я перечислю некоторых из них и призываю вас самостоятельно исследовать экосистему.

Chaco

Пакет Chaco (*http://code.enthought.com/chaco/*), разработанный компанией Enthought, представляет собой инструментарий для построения графиков. Он

годится как для рисования статических графиков, так и для интерактивной визуализации. Особенно хорош он, когда нужно выразить сложные визуализации, характеризуемые наличием взаимозависимостей между данными. По сравнению с matplotlib Chaco гораздо лучше поддерживает взаимодействие с элементами графика, а отрисовка производится очень быстро, так что этот пакет будет хорошим выбором для построения интерактивных приложений с графическим интерфейсом.

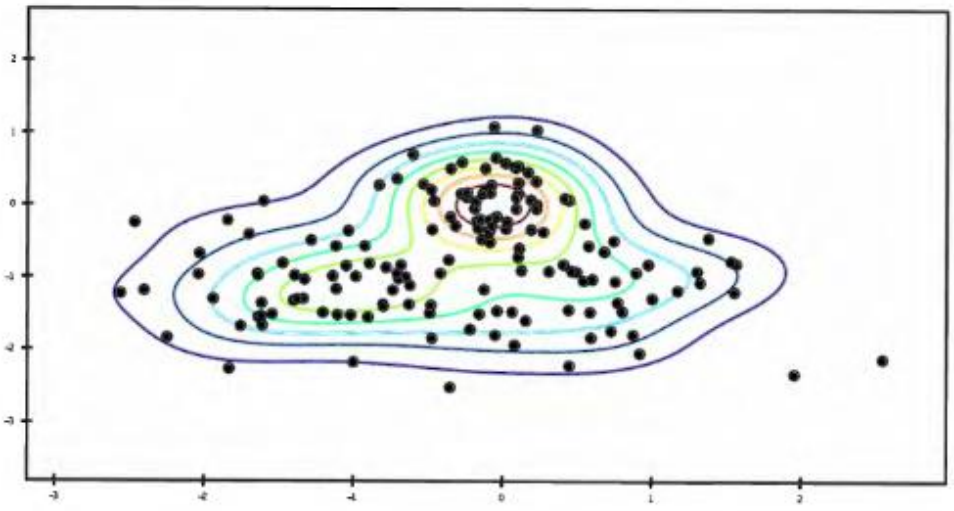

Рис. 8.26. Пример графика, построенного Chaco

mayavi

Проект mayavi, разработанный Прабху Рамачандраном (Prabhu Ramachandran) Гаэлем Вароко (Gaël Varoquaux) и другими, – это библиотека трехмерной графики, построенная поверх написанной на C++ библиотеки VTK с открытым исходным кодом. mayavi, как и matplotlib, интегрирована с IPython, поэтому с ней легко работать интерактивно. Графики можно панорамировать, поворачивать и масштабировать с помощью мыши и клавиатуры. Я использовал mayavi для создания од-

ной иллюстрации укладывания в главе 12. И хотя я не привожу здесь свой код работы с mayavi, в сети достаточно документации и примеров. Я полагаю, что во многих случаях это неплохая альтернатива таким технологиям, как WebGL, хотя интерактивными графиками труднее поделиться.

Прочие пакеты

Разумеется, для Python хватает и других библиотек и приложений для визуализации: PyQwt, Veusz, gnuplot-py, biggles и т. д. Я видел, как удачное применение PyQwt в графических приложениях, написанных с помощью каркаса Qt и надстройки PyQt над ним. Многие библиотеки продолжают активно развиваться (некоторые являются частью гораздо более крупных приложений), но в последние годы наблюдается общая тенденция к дрейфу в сторону веб-технологий и отхода от графики для настольных компьютеров. Я еще скажу об этом несколько слов в следующем разделе.

Будущее средств визуализации

Похоже, будущее за веб-технологиями (на основе JavaScript), и от этого никуда не деться. Без сомнения, за прошедшие годы вы встречались с самыми разными видами статических и интерактивных визуализаций, написанных на Flash или JavaScript. Постоянно появляются все новые и новые инструменты (например, d3.js и многочисленные отпочковавшиеся от него проекты). Напротив, разработка средств визуализации на платформах, отличных от веб, в последние годы резко замедлилась. Это относится как к Python, так и к другим средам для анализа данных

Требования к представлению и оформлению результатов самостоятельной работы

Самостоятельная работа включает в себя повторение теоретического и практического материала дисциплины, заслушиваемого и конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные проблемные вопросы по темам занятий; самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со специальным программным обеспечением.

Результаты самостоятельной работы представляются и оформляются в виде ответов на основные положения теоретического и практического материала дисциплины по темам; письменного разбора процесса решения практических заданий и задач; собственных действий, осуществляемых в ходе выполнения лабораторных работ.

В случае подготовки слайдов для защиты проекта, они должны быть контрастными (рекомендуется черный цвет шрифта на светлом фоне), кегль текста слайдов – не менее 22pt, заголовков – 32pt. Основная цель использования слайдов - служить вспомогательным инструментом к подготовленному выступлению, цитирование больших фрагментов текста на слайдах не допускается. Приветствуется использование рисунков, графиков, таблиц, интерактивного материала, однако, следует предусмотреть выбор цвета и толщину линий.

Слайды должны содержать титульный лист, цели и задачи (не более 2-х слайдов с обзором актуальности, новизны, теоретической и практической значимости работы), основные публикации с их кратким обзором (1-2 слайда), формальную постановку задачи и формулировку моделей (1-2 слайда), краткое тезисное (!) изложение ключевых положений работы (разумное количество слайдов с учетом общего времени выступления), заключение (с изложением результатов работы, подведением выводов, обсуждением практического использования работы, возможностей проведения дальнейших исследований и разработок в данной области).

Как правило, 12-15 слайдов оказывается достаточным для полного представления работы.

Критерии оценки выполнения самостоятельной работы

Общие критерии оценки выполнения самостоятельной работы – правильность ответов на вопросы по темам теоретической части дисциплины, верность получаемых ответов в ходе решения практических заданий и задач,

достижение правильного результата при осуществлении собственных действий по лабораторным работам.

Оценивание знаний в форме собеседования проводится по критериям:

- логичность изложения, знание и понимание основных аспектов и дискуссионных проблем по теме;
- владение методами и приемами анализа теоретических и/или практических аспектов по теме.

Оценивание знаний в форме проекта проводится по критериям:

- завершенность и полнота выполненных заданий в рамках работы;
- владение методами и приемами решения конкретных задач и самостоятельность использования специализированного программного обеспечения;
- качество оформления письменного отчета в соответствии с правилами и стандартами оформления.