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

Факультет инновационных технологий

УТВЕРЖДЕНО:
Декан
С. В. Шидловский

Оценочные материалы по дисциплине

Прикладной искусственный интеллект

по направлению подготовки / специальности

09.03.02 Информационные системы и технологии

Направленность (профиль) подготовки/ специализация:
Программное и аппаратное обеспечение беспилотных авиационных систем

Форма обучения
Очная

Квалификация
Инженер - программист
Инженер - разработчик

Год приема
2024

СОГЛАСОВАНО:
Руководитель ОП
С. В. Шидловский

Председатель УМК
О.В. Вусович

Томск – 2024

1. Компетенции и индикаторы их достижения, проверяемые данными оценочными материалами

Целью освоения дисциплины является формирование следующих компетенций:
ПК-1 Способен разрабатывать ПО для интеллектуального управления БАС.

Результатами освоения дисциплины являются следующие индикаторы достижения компетенций:
РОПК-1. Знает принципы разработки ПО для интеллектуального управления БАС.
РОПК-1.2 Умеет осуществлять обучение нейронных сетей на заранее определенных данных.
РОПК-1.3 Умеет осуществлять реализацию обученных нейронных сетей в программном коде.
РОПК-1.6 Умеет осуществлять реализацию алгоритмов обработки изображений в программном коде.

2. Оценочные материалы текущего контроля и критерии оценивания

Элементы текущего контроля:
– тесты;
– отчёт по лабораторным работам.

Тест (Ограничение по времени 1,5 часа, автоматическая проверка ответов, 25 вопросов, *РОПК-1.2*)

Примеры вопросов:

1. Что такое Dropout в контексте обучения нейронных сетей?

а) Процесс полной остановки обучения модели
б) Метод регуляризации, где случайным образом отключаются нейроны во время обучения
в) Механизм увеличения количества слоев в сети
г) Техника увеличения размера входных данных

2. Для какого типа задач обычно используются сверточные нейронные сети (CNN)?

а) Анализ временных рядов
б) Прогнозирование финансовых рынков
в) Создание рекомендательных систем
г) Распознавание образов и изображений

Ключи: 1 б), 2 г).
Критерии оценивания: тест считается пройденным, если студент верно ответил на 20 и более вопросов.

Лабораторная работа (РОПК-1.3, РОПК-1.6)

Примеры задач:
Задача 1
1) Подготовить данные для анализа, включая преобразование категориальных признаков в числовые и удаление пропущенных значений. 2) Разделить данные на обучающую и тестовую выборки в соотношении 80% к 20% соответственно. 3) Создать модели дерева решений и градиентного бустинга, проверить их точность на тестовой выборке. 4) Для моделей подобрать оптимальные параметры (например, глубину дерева или количество деревьев в ансамбле), при которых достигается максимальная точность работы. 5) Отобразить классы, для которых модель совершает ошибки.
Задача 2

выбрать из представленных ниже архитектур одну и реализовать её в файле model.py. Архитектуры отражены на изображениях. Изображения содержат информацию о: количестве свёрточных и полносвязных слоёв; количестве каналов для карт признаков, полученных на выходе свёрточных слоёв; параметров для свёртки и пулинга; функциях активации, выполняемых после свёрточных слоёв.

Ответы:
Задача 1.

```
import numpy as np
import pandas as pd
from pathlib import Path
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics
p = Path('.')
penguins = pd.read_csv('penguins.csv')
penguins['male'] = penguins['sex'].apply(lambda x: 1 if x == 'male' else 0)
penguins['female'] = penguins['sex'].apply(lambda x: 1 if x == 'female' else 0)
penguins = penguins.drop('sex', axis=1)
island_columns = ['Torgersen', 'Biscoe', 'Dream']
for island in island_columns:
   penguins[island] = penguins['island'].apply(lambda x: 1 if x == island else 0)
penguins = penguins.drop('island', axis=1)
classes = penguins['species'].unique()
class_mapping = {classes[i]: i+1 for i in range(len(classes))}
penguins['species'] = penguins['species'].map(class_mapping)
penguins_cleaned = penguins.dropna()
x_train, x_test, y_train, y_test = train_test_split(
   np.array(penguins_cleaned.iloc[:,1:]),
   np.array(penguins_cleaned['species']),
   test_size = 0.2,
   random_state=)
tree_classifier = DecisionTreeClassifier(max_depth=3)
tree_classifier.fit(x_train, y_train)
predictions_tree = tree_classifier.predict(x_test)
score_tree = tree_classifier.score(x_test, y_test)
print("Точность модели дерева решений:", score_tree)
cm_tree = metrics.confusion_matrix(y_test, predictions_tree,
   labels=np.unique(penguins_cleaned['species']))
print("Матрица ошибок для дерева решений:\n", cm_tree)
disp_tree = metrics.ConfusionMatrixDisplay(confusion_matrix=cm_tree,
   display_labels=classes)
disp_tree.plot()
plt.title("Confusion Matrix for Decision Tree")
plt.show()
boosting_classifier = GradientBoostingClassifier(n_estimators=100, max_depth=3)
boosting_classifier.fit(x_train, y_train)
predictions_boosting = boosting_classifier.predict(x_test)
```

```
score_boosting = boosting_classifier.score(x_test, y_test)
print("Точность модели градиентного бустинга:", score_boosting)
cm_boosting = metrics.confusion_matrix(y_test, predictions_boosting,
   labels=np.unique(penguins_cleaned['species']))
print("Матрица ошибок для градиентного бустинга:\n", cm_boosting)
disp_boosting = metrics.ConfusionMatrixDisplay(confusion_matrix=cm_boosting,
   display_labels=classes)
disp_boosting.plot()
plt.title("Confusion Matrix for Gradient Boosting")
plt.show()
from sklearn.model_selection import GridSearchCV
param_grid_tree = {'max_depth': range(1, 10)}
grid_search_tree = GridSearchCV(DecisionTreeClassifier(), param_grid_tree, cv=5)
grid_search_tree.fit(x_train, y_train)
best_depth_tree = grid_search_tree.best_params_['max_depth']
print(f"Оптимальная глубина дерева решений: {best_depth_tree}")
param_grid_boosting = {'n_estimators': range(10, 150, 10), 'max_depth': range(1, 10)}
grid_search_boosting = GridSearchCV(GradientBoostingClassifier(),
   param_grid_boosting, cv=5)
grid_search_boosting.fit(x_train, y_train)
best_params_boosting = grid_search_boosting.best_params_
print(f"Оптимальные параметры для градиентного бустинга:
   {best_params_boosting}")
```

Задача 2.

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class CNN(nn.Module):
   def __init__(self, num_classes=10):
      super(CNN, self).__init__()
      self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
      self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
      self.conv3 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
      self.conv4 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
      self.conv5 = nn.Conv2d(256, 512, kernel_size=3, padding=1)
      self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
      self.fc1 = nn.Linear(512 * 4 * 4, 1024)
      self.fc2 = nn.Linear(1024, 512)
      self.fc3 = nn.Linear(512, num_classes)

   def forward(self, x):
      x = F.relu(self.conv1(x))
      x = self.pool(x)
      x = F.relu(self.conv2(x))
      x = self.pool(x)
      x = F.relu(self.conv3(x))
      x = self.pool(x)
      x = F.relu(self.conv4(x))
      x = self.pool(x)
```

```
        x = F.relu(self.conv5(x))
        x = self.pool(x)
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

3. Оценочные материалы итогового контроля (промежуточной аттестации) и критерии оценивания

Экзамен проводится в письменной форме по билетам. Экзаменационный билет состоит из двух частей. Продолжительность экзамена 1,5 часа.

Первая часть экзаменационного билета представляет собой 1 вопрос, проверяющий РОПК-1.2. Ответ на вопрос первой части дается в развернутой форме.

Вторая часть содержит один вопрос, оформленный в виде практической задачи, проверяющий РОПК-1.3 и РОПК-1.6. Ответ на вопрос второй части предполагают решение задачи и краткую интерпретацию полученных результатов.

Примерный перечень вопросов первой части экзаменационного билета:

1) Искусственный интеллект (ИИ), слабый ИИ, сильный ИИ.
2) Экспертная система. Задача регрессии. Задача классификации.
3) Машинное обучение, виды машинного обучения. Модель машинного обучения, этапы получения обученной модели, различие между гиперпараметрами и параметрами модели.
4) Типовые задачи при подготовке данных, определение ошибок и выбросов в данных.
5) Набор данных (датасет). Функция потерь, средняя ошибка, средняя абсолютная ошибка, среднеквадратическая ошибка, средняя абсолютная относительная ошибка.
6) Метод опорный векторов. Метод k-ближайших соседей.
7) Дерево решений, гиперпараметры деревьев решений.
8) Случайный лес.
9) Градиентный бустинг деревьев решений.
10) Математическая модель нейрона. Полносвязный слой. Функция активации. Фазы работы нейронной сети при обучении. Основные типы задач, которые решают нейронные сети.
11) Глубокое обучение. Явление переобучения, способы борьбы с переобучением.Кросс-валидация.
12) Сверточные нейронные сети, гиперпараметры сверточного слоя.
13) Линейная регрессия. Логистическая регрессия. Полностью сверточные сети.
14) Градиентный спуск. Autoencoder (автокодер, автоэнкодер, AE).

Примерный перечень вопросов второй части экзаменационного билета:

1) Создать алгоритм классификации методом k-ближайших соседей;
2) Построить пример линейной предсказательной модели машиного обучения;
3) Построить структурную схему и записать математическую модель перцептрона;
4) Построить структурную схему полносвязной нейронной сети, содержащей 3 скрытых слоя по 5 нейронов каждый, а также решающей задачу бинарной классификации входных данных.
5) Построить структурную схему сверточной нейронной сети

В таблице 1 приведены критерии оценивания ответов на экзаменационный билет.

Таблица 1 - Критерии оценивания ответов на экзаменационный билет

Характеристика ответов на экзаменационный билет	Оценка
Получены развернутые ответы по двум частям экзаменационного билета	«отлично»
Получен развернутый ответ по одной части и краткий ответ по второй части экзаменационного билета	«хорошо»
Получен только развернутый ответ по одной части экзаменационного билета	«удовлетворительно»
Отсутствует развернутый ответ по обеим частям экзаменационного билета	«неудовлетворительно»

В случае, если в течение курса студент не присутствовал на занятиях, то в течение времени, отведенного на проведение экзамена, у него есть возможность пройти тест из 15 вопросов, сдать 5 лабораторнях заданий с отчетами, сдать экзамен и получить итоговую оценку.

4. Оценочные материалы для проверки остаточных знаний (сформированности компетенций)

Пример вопросов теста:

1. Что такое переобучение в машинном обучении? (РОПК-1.2.)

а) ситуация, когда модель слишком сложная и не может обобщать данные
б) метод обучения, при котором модель адаптируется к новым данным
в) подход к обучению, основанный на постоянном добавлении новых данных
г) алгоритм оптимизации, используемый для улучшения точности модели

2. Какой тип нейронной сети предназначен для решения задачи классификации изображений? (РОПК-1.6.)

а) рекурентная нейронная сеть
б) полносвязная нейронная сеть
в) сверточная нейронная сеть
г) генеративная нейронная сеть

Ключи: 1 а), 2 в).

Пример задачи:

Задача (РОПК-1.3, РОПК-1.6.)

Загрузить датасет CIFAR-10 и разделить данные на обучающую и тестовую выборки в соотношении 80% к 20%. Создать простую сверточную нейронную сеть (CNN) для классификации изображений, состоящую из нескольких слоев: сверточный слой, функция активации ReLU, слой подвыборки (max-pooling), несколько полносвязных слоев. Обучить модель на обучающей выборке и оценить точность модели на тестовой выборке. Подобрать гиперпараметры сети (например, количество эпох обучения, скорость обучения, количество фильтров в сверточных слоях) для достижения максимальной точности.

Ответ:

```
import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torch.optim as optim
transform = transforms.Compose([transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True,
    transform=transform)
```

```
trainloader = torch.utils.data.DataLoader(trainset, batch_size=4, shuffle=True,
   num_workers=2)
testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True,
   transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=4, shuffle=False,
   num_workers=2)
classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
class Net(nn.Module):
   def __init__(self):
      super(Net, self).__init__()
      self.conv1 = nn.Conv2d(3, 6, 5)
      self.pool = nn.MaxPool2d(2, 2)
      self.conv2 = nn.Conv2d(6, 16, 5)
      self.fc1 = nn.Linear(16 * 5 * 5, 120)
      self.fc2 = nn.Linear(120, 84)
      self.fc3 = nn.Linear(84, 10)
   def forward(self, x):
      x = self.pool(torch.relu(self.conv1(x)))
      x = self.pool(torch.relu(self.conv2(x)))
      x = x.view(-1, 16 * 5 * 5)
      x = torch.relu(self.fc1(x))
      x = torch.relu(self.fc2(x))
      x = self.fc3(x)
      return x
net = Net()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
for epoch in range(2):
    running_loss = 0.    for i, data in enumerate(trainloader, 0):
      inputs, labels = data
      optimizer.zero_grad()
      outputs = net(inputs)
      loss = criterion(outputs, labels)
      loss.backward()
      optimizer.step()
      running_loss += loss.item()
      if i % 2000 == 1999:
         print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss / 2000))
         running_loss = 0.print('Обучение завершено')
correct = total = with torch.no_grad():
   for data in testloader:
      images, labels = data
      outputs = net(images)
      _, predicted = torch.max(outputs.data, 1)
      total += labels.size(0)
      correct += (predicted == labels).sum().item()
print('Точность сети на тестовых изображениях: %d %%' % (100 * correct / total))
```

5. Информация о разработчиках

Бондарчук Антон Сергеевич, кандидат технических наук, доцент кафедры информационного обеспечения интеллектуальной деятельности факультета инновационных технологий.