Жарқын Ақтілек

10-есеп

Робототехникадағы оптимизация :Apm тапсырмасында (Reinforcement Learning) Adam және SGD-ның әсерін тексер

import gym

import torch

from stable\_baselines3 import PPO

from stable\_baselines3.common.evaluation import evaluate\_policy

def train\_and\_evaluate(optimizer, optimizer\_name):

env = gym.make("CartPole-v1")

policy\_kwargs = dict(optimizer\_class=optimizer)

model = PPO("MlpPolicy", env, policy\_kwargs=policy\_kwargs, verbose=0)

model.learn(total\_timesteps=10000)

mean\_reward, \_ = evaluate\_policy(model, env, n\_eval\_episodes=10)

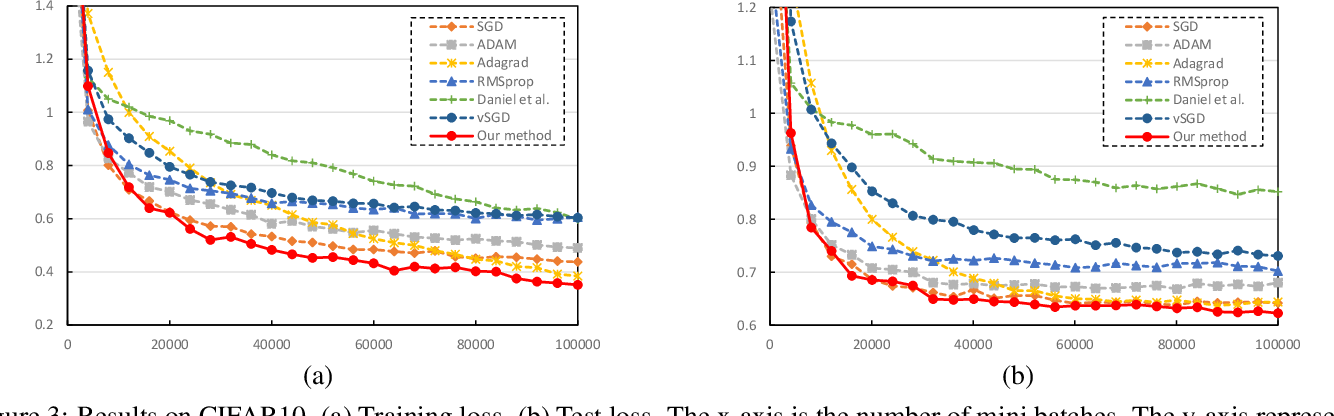
print(f"Optimizer: {optimizer\_name}, Mean Reward: {mean\_reward}")

env.close()

# Запуск обучения с Adam и SGD

train\_and\_evaluate(torch.optim.Adam, "Adam")

train\_and\_evaluate(torch.optim.SGD, "SGD")



11 –Есеп

L1 және L2 регулярилациясының оптимизатрларға әсері: Әртүрлі оптимизаторлармен L1 және L2 регуляризациясының рөлін математикалық және грфикалық анықтау

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

# Жасанды деректер

np.random.seed(42)

X = np.linspace(-1, 1, 100).reshape(-1, 1)

y = 3 \* X.squeeze() + np.random.normal(0, 0.1, X.shape[0])

# Негізгі модель құру функциясы

def create\_model(regularizer=None, optimizer='adam'):

model = tf.keras.Sequential([

tf.keras.layers.Dense(10, activation='relu', kernel\_regularizer=regularizer),

tf.keras.layers.Dense(1)

])

model.compile(optimizer=optimizer, loss='mse')

return model

# Әртүрлі модельдер (L1, L2 және Regularizationсыз)

optimizers = ['adam', 'sgd']

regularizers = [None, tf.keras.regularizers.l1(0.01), tf.keras.regularizers.l2(0.01)]

regularizer\_names = ['No Regularization', 'L1 Regularization', 'L2 Regularization']

# График үшін дайындық

plt.figure(figsize=(10, 5))

for i, optimizer in enumerate(optimizers):

plt.subplot(1, 2, i+1)

for regularizer, name in zip(regularizers, regularizer\_names):

model = create\_model(regularizer, optimizer)

history = model.fit(X, y, epochs=100, verbose=0, batch\_size=10)

plt.plot(history.history['loss'], label=f'{name}')

plt.title(f'Optimizer: {optimizer}')

plt.xlabel('Epochs')

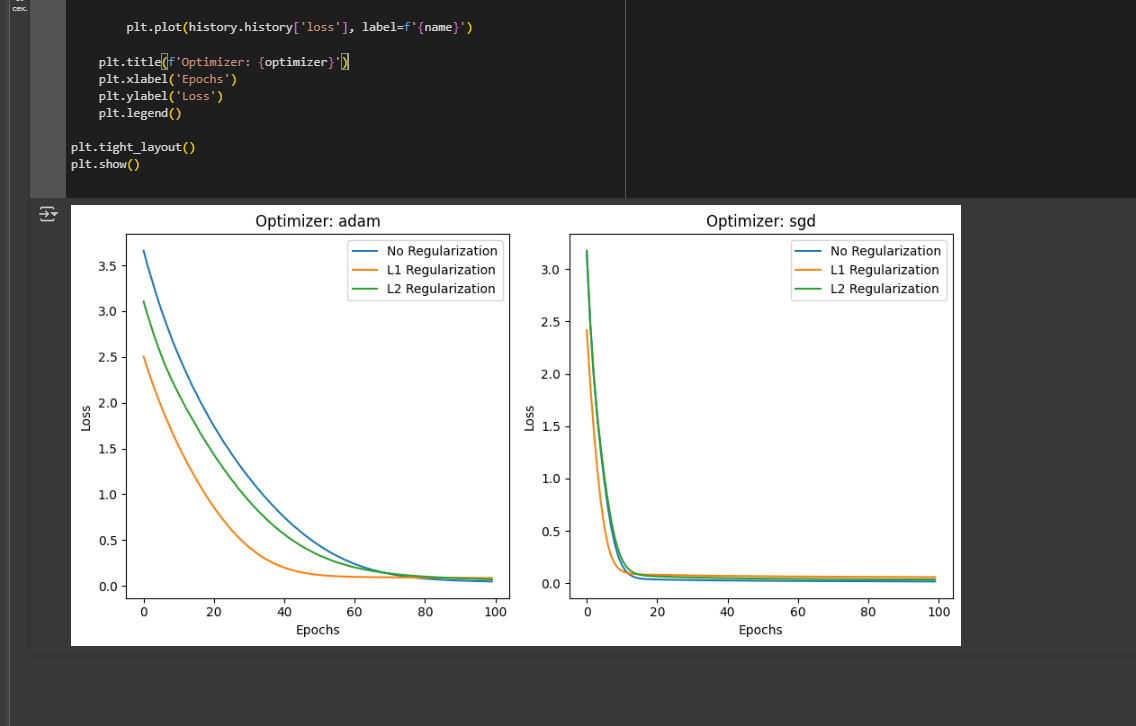
plt.ylabel('Loss')

plt.legend()

plt.tight\_layout()

plt.show()

Нәтижесі



12-есеп

Оверфитингте қарсы оптимизация стратегилары: Dropout және Weight Decay әдістерін қолданып, олардың Adam және RMSprop-пен бірге жұмыс істеуін бағалаңыз.

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import numpy as np

import matplotlib.pyplot as plt

# Генерация синтетикалық деректер

np.random.seed(42)

x\_train = np.linspace(-1, 1, 500)

y\_train = 3 \* x\_train \*\* 2 + 2 + np.random.normal(0, 0.1, size=x\_train.shape)

x\_train = x\_train.reshape(-1, 1)

y\_train = y\_train.reshape(-1, 1)

# Модель құру функциясы

def build\_model(optimizer, use\_dropout=False, weight\_decay=0.0):

model = keras.Sequential()

model.add(layers.Dense(128, activation='relu', kernel\_regularizer=keras.regularizers.l2(weight\_decay)))

if use\_dropout:

model.add(layers.Dropout(0.3))

model.add(layers.Dense(64, activation='relu', kernel\_regularizer=keras.regularizers.l2(weight\_decay)))

if use\_dropout:

model.add(layers.Dropout(0.3))

model.add(layers.Dense(1))

model.compile(optimizer=optimizer, loss='mse')

return model

# Оптимизаторларды салыстыру

optimizers = {

'Adam': keras.optimizers.Adam(learning\_rate=0.01),

'RMSprop': keras.optimizers.RMSprop(learning\_rate=0.01)

}

results = {}

for opt\_name, optimizer in optimizers.items():

for dropout in [False, True]:

for wd in [0.0, 0.01]:

model = build\_model(optimizer, use\_dropout=dropout, weight\_decay=wd)

history = model.fit(x\_train, y\_train, epochs=100, verbose=0, batch\_size=32)

key = f"{opt\_name}\_dropout\_{dropout}\_wd\_{wd}"

results[key] = history.history['loss']

# Нәтижелерді визуализациялау

plt.figure(figsize=(12, 6))

for key, loss in results.items():

plt.plot(loss, label=key)

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.title("Dropout және Weight Decay әсері")

plt.show()

Нәтижесі