

Torch Models Example

Torch Models Example

Файлы: `examples/torch_models/train.py` и `predict.py`.

“ **Проверено на локальном Docker-стеке:** оба скрипта отработывают до конца. Недостающие данные (см. ниже) добавлены в репозиторий.

?????????: ????????????
??????

Оба скрипта читают `examples/torch_models/data/data.csv`:

```
df = pd.read_csv(data_dir / "data.csv", sep=";")
X = df.drop("class", axis=1).values
y = df["class"].values
```

Папки `examples/torch_models/data/` в репозитории не было — файл данных отсутствовал, запуск падал с `FileNotFoundError`. Добавлен `examples/torch_models/data/data.csv` — синтетический датасет из 180 строк, 3 класса (по 60 строк каждый), 4 числовых признака (`f1`...`f4`, гауссовы "блобы" вокруг разных центров) и колонка `class` (0/1/2), разделитель `;`. Уже обученные веса (`models/model.pt`, `models/model_automl.pt`) в репозитории тоже есть и теперь соответствуют этим данным — оба файла перезаписываются при каждом запуске `train.py`.

train.py

```
class SimpleNet(nn.Module):
    def __init__(self, input_dim, num_classes):
        super().__init__()
        self.fc1 = nn.Linear(input_dim, 512)
```

```

self.dropout1 = nn.Dropout(0.3)
self.fc2 = nn.Linear(512, 256)
self.dropout2 = nn.Dropout(0.3)
self.fc3 = nn.Linear(256, num_classes)

# forward: fc1 -> ReLU -> Dropout -> fc2 -> ReLU -> Dropout -> fc3

wrapper = TorchModelWrapper(model)
wrapper.learn(X_train=X, y_train=y, epochs=20, batch_size=4, lr=1e-3)
wrapper.save(models_dir / "model.pt")

wrapper_aml = TorchModelWrapper.auto_search(X, y)
wrapper_aml.save(models_dir / "model_automl.pt")

```

Сначала обучает вручную заданную трёхслойную сеть (512→256→classes, dropout 0.3), сохраняет как `model.pt`. Затем отдельно прогоняет `TorchModelWrapper.auto_search(X, y)` — полный перебор 54 архитектур (см. страницу Torch Models в книге «Документация MLumina») — и сохраняет лучшую как `model_automl.pt`. Обратите внимание: `batch_size=4` в ручном обучении — очень маленький батч, подходит только для небольших демонстрационных датасетов.

predict.py

Пересоздаёт **точно такую же** архитектуру `SimpleNet` (руками, не через сериализацию структуры — `TorchModelWrapper.save` сохраняет только `state_dict`, не саму архитектуру), загружает веса `model.pt` и делает предсказания:

```

model = SimpleNet(input_dim, num_classes)
state = torch.load(models_dir / "model.pt", map_location="cpu")
model.load_state_dict(state)

wrapper = TorchModelWrapper(model, device="cpu")
probs = wrapper.predict(X, return_probs=True)
labels = wrapper.predict(X, return_probs=False)

```

Важно: `input_dim/num_classes` в `predict.py` вычисляются заново из CSV — если структура данных при инференсе отличается от той, на которой обучалась модель (другое число признаков/классов), `load_state_dict` упадёт из-за несовпадения размерностей слоёв. Здесь нет сохранения/восстановления архитектуры "из коробки" — это ограничение `TorchModelWrapper.save`, а не только примера.