

Ray Wrapper: Linear Regression and SVM

Ray Wrapper: Linear Regression and SVM

Самые простые примеры из `ray_wrapper/examples/` — хорошая отправная точка, чтобы понять паттерн `ray.tune.Tuner`, используемый во всех остальных моделях пакета `mlumina.ray_wrapper`. У обеих моделей есть только `tune.py` (готовой функции `train_*` без подбора гиперпараметров для них не предусмотрено).

Linear Regression —

`ray_wrapper/examples/lin_reg/tune.py`

```
ray.init(address="auto")

search_space = {
    "fit_intercept": tune.choice([True, False]),
    "positive": tune.choice([True, False]),
    "test_size": 0.2,
    "random_state": 42,
}

tune_lin_reg(search_space=search_space, num_samples=4, resources={"CPU": 1},
max_concurrent_trials=4)
```

Внутри (`ray_wrapper/models/lin_reg/lin_reg.py`) каждый trial строит sklearn-датасет `diabetes` (`build_sklearn_ray_datasets`, регрессия), обучает `sklearn.linear_model.LinearRegression` с параметрами из `search_space`, репортит `rmse/r2`. Отбор — `ASHAScheduler(metric="rmse",`

`mode="min")`. `test_size`/`random_state` передаются как обычные (не варьируемые) значения конфига — `tune.choice`/`tune.loguniform` применяются только к `fit_intercept`/`positive`.

SVM —

`ray_wrapper/examples/svm/tune.py`

```
ray.init(address="auto")

search_space = {
    "kernel": tune.choice(["rbf", "linear", "poly"]),
    "C": tune.loguniform(1e-2, 1e2),
    "gamma": tune.choice(["scale", "auto"]),
    "degree": tune.choice([2, 3, 4]),
    "class_weight": tune.choice([None, "balanced"]),
    "test_size": 0.2,
    "random_state": 42,
}

tune_svm(search_space=search_space, num_samples=8, resources={"CPU": 1},
max_concurrent_trials=4)
```

В отличие от `lin_reg`, это **классификация**: внутри (`ray_wrapper/models/svm/svm.py`) каждый `trial` строит датасет `breast_cancer` (бинарная классификация), обучает `Pipeline(StandardScaler → SVC(kernel, C, gamma, degree, class_weight))`, репортит `accuracy`. Отбор — `ASHAScheduler(metric="accuracy", mode="max")`.

??? ??????????

Оба скрипта вызывают `ray.init(address="auto")` — рассчитаны на запуск **изнутри** кластера (в контейнере `head`/`worker`, где переменная окружения `RAY_ADDRESS`/сокет автообнаружения кластера уже настроены), либо нужно подключаться явно через `ray://<head>:10001`, поправив скрипт:

```
# из контейнера head/worker
python ray_wrapper/examples/lin_reg/tune.py
python ray_wrapper/examples/svm/tune.py
```

Результаты (`tune.Tuner.fit()` → `ResultGrid`) сохраняются локально в `/tmp/ray_results/tune_lin_reg_<uuid>` / `tune_svm_<uuid>` на том узле, где физически исполнялся

driver-процесс trial'a — при работе на многих worker'ах это может быть не тот узел, с которого был запущен скрипт.

Revision #1

Created 2026-09-23 06:54:47 UTC by Маслов Алексей

Updated 2026-09-23 06:54:47 UTC by Маслов Алексей