НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет ПИиКТ

Системы искусственного интеллекта

Лабораторная работа № 3

Выполнил студент

Набокова Алиса Владиславовна

Группа № P3320

Преподаватель: Королёва Юлия Александровна

г. Санкт-Петербург

2025

Задание:

Реализовать класс модели SVM для классификации с обучением по батчам на датасете moabb.datasets.BI2013a

Обучить модель и вывести метрики качества (ошибки, точность предсказаний, и тд)

Чтобы приступить к обучению модели необходимо ознакомиться с содержимым датасета. В нем представлен эксперимент парадигмы P300. На рисунках 1-2 представлены графики событийно-связанных потенциалов электрических сигналов мозга субъекта 24 на стимул по всем каналам и по каждому каналу отдельно. А также листинг 1-2 отображает исходный код приведенных графиков

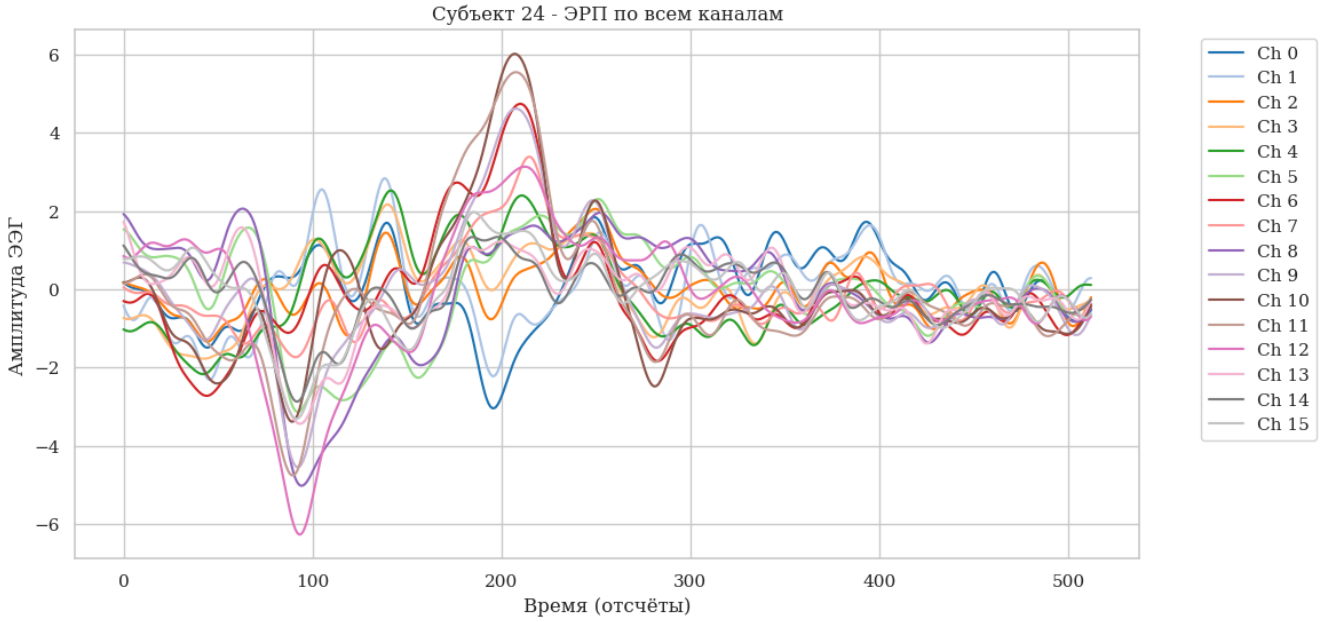

Рисунок 1. ERP субъекта 24 по всем каналам

Листинг 1. Код ERP графика субъекта 24 по всем каналам

```
target_trials = X_subj[y_subj == 'Target']

nontarget_trials = X_subj[y_subj == 'NonTarget']


mean_target = np.mean(target_trials, axis=0)

mean_nontarget = np.mean(nontarget_trials, axis=0)


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


colors = plt.colormaps.get_cmap('tab20')


for ch in range(X_subj.shape[1]):

   plt.plot(mean_target[ch], label=f'Ch {ch}', color=colors(ch))


plt.title(f'Субъект {subj} - ERP по всем каналам')

plt.xlabel('Время (отсчёты)')

plt.ylabel('Амплитуда ЭЭГ')

plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

plt.show()
```

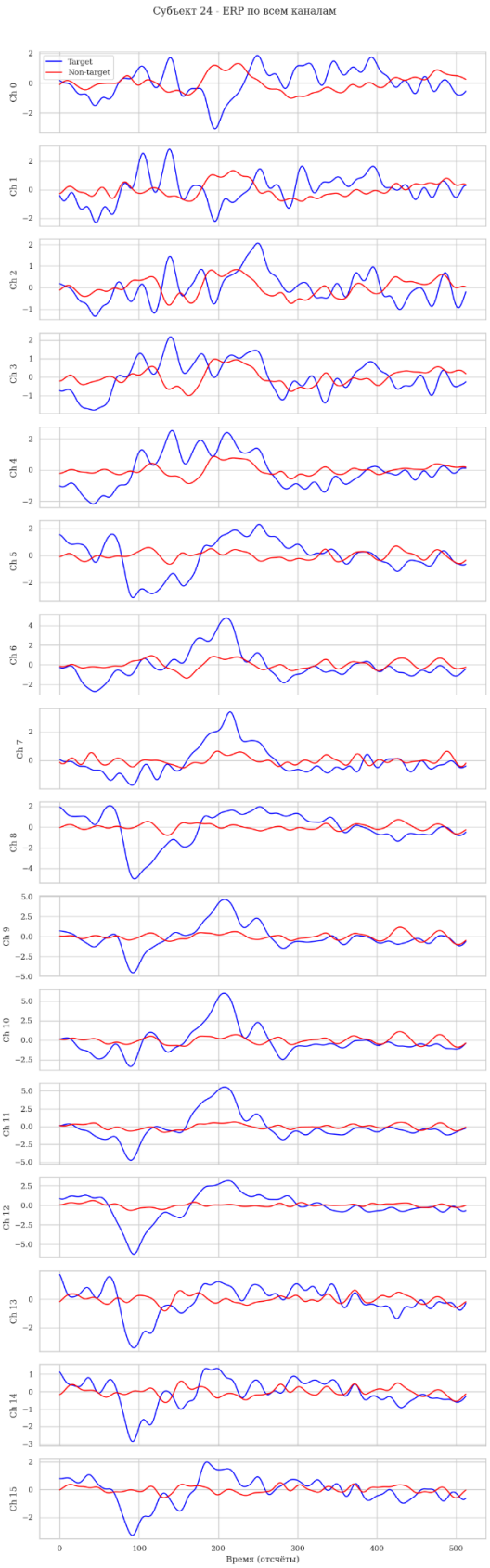

Рисунок 2. ERP субъекта 24 по каждому каналу

Листинг 2. Код ERP графика субъекта 24 по каждому каналу

```
n_channels = X_subj.shape[1]

fig, axes = plt.subplots(n_channels, 1, figsize=(10, 2*n_channels),
sharex=True)


for ch in range(n_channels):

   axes[ch].plot(mean_target[ch], label='Target', color='blue')

   axes[ch].plot(mean_nontarget[ch], label='Non-target', color='red')

   axes[ch].set_ylabel(f'Ch {ch}')

   if ch == 0:

       axes[ch].legend()


axes[-1].set_xlabel('Время (отсчёты)')

plt.suptitle(f'Субъект {subj} - ERP по всем каналам')

plt.tight_layout(rect=[0, 0, 1, 0.97])

plt.show()
```

По построенным графикам можно наглядно увидеть, где возникает парадигма P300 - то есть электрический сигнал, который возникает на некий стимул примерно на 300 мс после демонстрации стимула респонденту. Самый большой положительный пик это и есть P300. По графику видно, что он приходится на 10 и 11 каналы. Поэтому именно они наиболее важны при интерпретации результатов. На них наглядно видно, что целевой сигнал сильно выше других и имеет ярко выраженный пик. На участках мозга, к которым подключены каналы.

Далее переходим к подготовке данных и обучении модели SVM. На листинге 3-4 представлен код стандартизации данных перед обучением и сам алгоритм обучения модели по методу опорных векторов.

Листинг 3. Стандартизация данных

```
y_binary = np.where(y_subj == 'Target', 1, -1)


n_trials, n_channels, n_times = X_subj.shape

X_flat = X_subj.reshape(n_trials, -1)


scaler = StandardScaler()

X_scaled = scaler.fit_transform(X_flat)
```

Листинг 4. Класс модели SVM

```
class SVMModel:

    def __init__(self, lr=0.001, C=0.001, epochs=100, batch_size=32):

        self.lr = lr

        self.C = C

        self.epochs = epochs

        self.batch_size = batch_size

        self.w = None

        self.b = None

        self.loss_history = []

        self.metrics = {}


    def fit(self, X, y):

        X_t = torch.tensor(X, dtype=torch.float)

        y_t = torch.tensor(y, dtype=torch.float).view(-1, 1)
```

```
        n_features = X.shape[1]
        self.w = torch.zeros(n_features, 1, dtype=torch.float,
requires_grad=True)
        self.b = torch.zeros(1, dtype=torch.float, requires_grad=True)

        dataset = TensorDataset(X_t, y_t)
        loader = DataLoader(dataset, batch_size=self.batch_size,
shuffle=True)

        print("-"*30 + "TRAINING" + "-"*30)
        for epoch in range(self.epochs):
            epoch_loss = 0.0
            for batch_X, batch_y in loader:
                scores = batch_y * (batch_X @ self.w + self.b)
                hinge = torch.clamp(1 - scores, min=0)
                loss = 0.5 * torch.sum(self.w ** 2) + self.C *
torch.mean(hinge)

                loss.backward()
                with torch.no_grad():
                    self.w -= self.lr * self.w.grad
                    self.b -= self.lr * self.b.grad
                    self.w.grad.zero_()
                    self.b.grad.zero_()

                epoch_loss += loss.item()

            avg_loss = epoch_loss / len(loader)
```

```
            self.loss_history.append(avg_loss)
            if (epoch + 1) % 10 == 0:
                print(f"Epoch: {epoch+1}/{self.epochs}, Loss:
{avg_loss:.4f}")

    def predict(self, X):
        X_t = torch.tensor(X, dtype=torch.float)
        with torch.no_grad():
            scores = X_t @ self.w + self.b
            return torch.sign(scores).numpy().flatten()

    def evaluate(self, X, y, dataset_name="Dataset"):
        y_pred = self.predict(X)
        y_true = y

        self.metrics[dataset_name] = {
            "Accuracy": accuracy_score(y_true, y_pred),
            "Precision": precision_score(y_true, y_pred, pos_label=1,
zero_division=0),
            "Recall": recall_score(y_true, y_pred, pos_label=1,
zero_division=0),
            "F1": f1_score(y_true, y_pred, pos_label=1, zero_division=0)
        }

    def get_metrics_table(self):
        print("-"*30 + "METRICS" + "-"*30)
        if not self.metrics:
            return pd.DataFrame()
        return pd.DataFrame(self.metrics).T
```

```
    def plot_loss(self):
        plt.figure(figsize=(8,5))
        sns.lineplot(x=range(len(self.loss_history)), y=self.loss_history,
color='hotpink')
        plt.title("SVM Training Loss")
        plt.xlabel("Epochs")
        plt.ylabel("Loss")
        plt.grid(True)
        plt.show()
```

Была выявлена особенность данных: выборка не сбалансированная

```
Уникальные метки: ['NonTarget' 'Target']
{np.str_('NonTarget'): np.int64(1200), np.str_('Target'): np.int64(240)}
```

Поэтому дополнительно анализируем метрики Precision и Recall

Далее на листинге 5 приведен код разделения данных на обучающую и тестовую выборки, обучение модели и вывод метрик качества обучения

Листинг 5. Код обучения модели SVM

```
X_train, X_test, y_train, y_test = train_test_split(
   X_scaled, y_binary, test_size=0.2, random_state=42, stratify=y_binary
)

model = SVMModel(lr=0.001, C=1.0, epochs=200, batch_size=32)
model.fit(X_train, y_train)

model.evaluate(X_train, y_train, dataset_name="Train")
model.evaluate(X_test, y_test, dataset_name="Test")

metrics_df = model.get_metrics_table()
print(metrics_df)

model.plot_loss()
```

Далее приведены метрики качества обучения модели: accuracy, precision, recall, f1-score, а также коэффициент средних потерь по батчам за эпоху и на рисунке 3 приведен график этих же потерь по эпохам

```
------------------------------TRAINING------------------------------
Epoch: 10/200, Loss: 0.5285
Epoch: 20/200, Loss: 0.4002
Epoch: 30/200, Loss: 0.3204
Epoch: 40/200, Loss: 0.2708
Epoch: 50/200, Loss: 0.2342
Epoch: 60/200, Loss: 0.2160
Epoch: 70/200, Loss: 0.2000
Epoch: 80/200, Loss: 0.1917
Epoch: 90/200, Loss: 0.1842
Epoch: 100/200, Loss: 0.1756
Epoch: 110/200, Loss: 0.1744
Epoch: 120/200, Loss: 0.1749
Epoch: 130/200, Loss: 0.1700
Epoch: 140/200, Loss: 0.1660
Epoch: 150/200, Loss: 0.1678
Epoch: 160/200, Loss: 0.1649
Epoch: 170/200, Loss: 0.1644
Epoch: 180/200, Loss: 0.1631
Epoch: 190/200, Loss: 0.1624
Epoch: 200/200, Loss: 0.1604
```

```
------------------------------METRICS-------------------------------

       Accuracy  Precision    Recall        F1

Train  0.975694   0.988095  0.864583  0.922222

Test   0.934028   0.871795  0.708333  0.781609
```

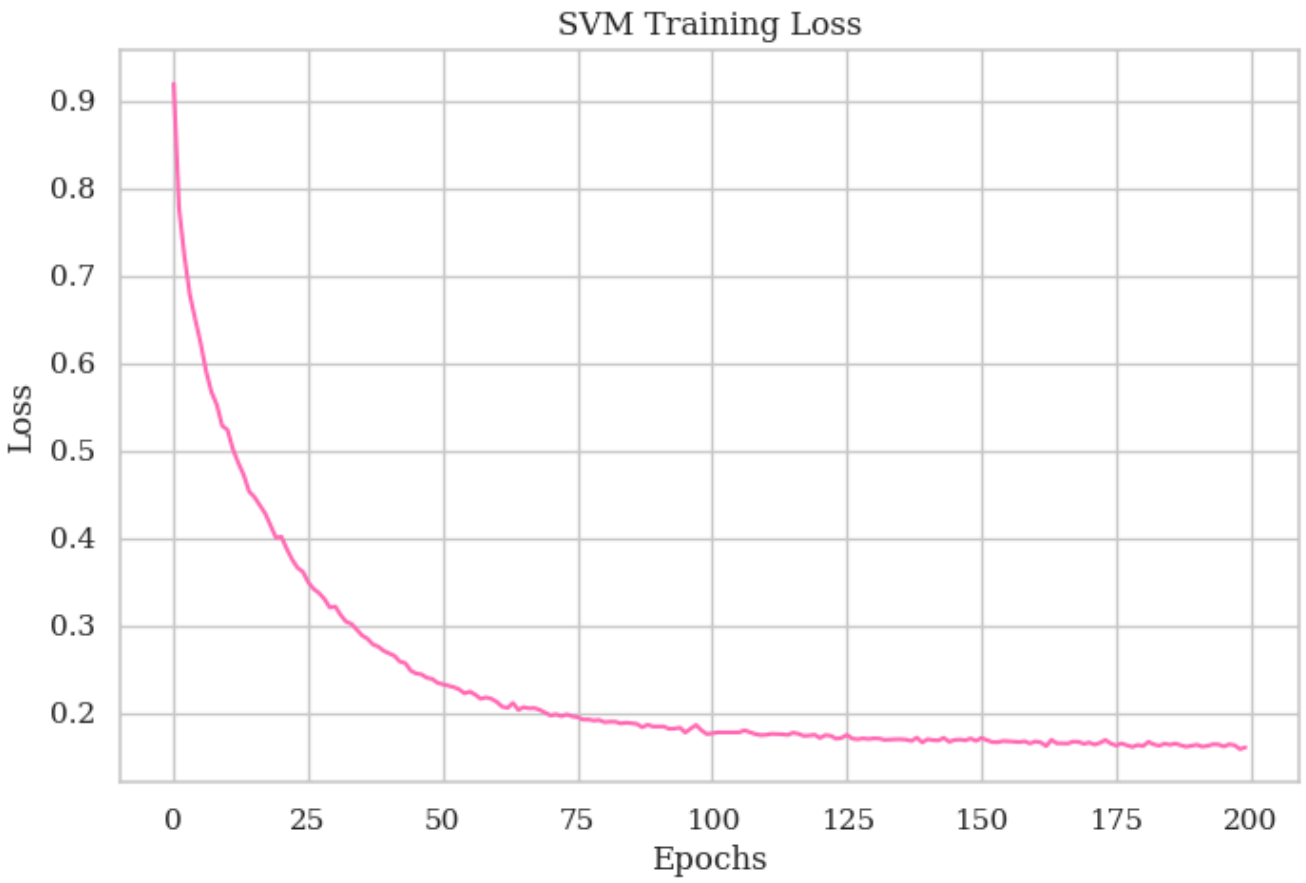

Рисунок 3. График потерь по эпохам

Вывод:

Анализ по Accuracy: Модель SVM показала высокую общую точность, как на тренировочных, так и на тестовых данных.

Анализ по Precision: На обучающей выборке модель почти не делает ложноположительных ошибок. На тесте precision ниже, но все еще достаточно высокий. Это говорит о том, что большинство предсказаний положительных объектов действительно являются положительными

Анализ по Recall: Модель не всегда удачно выявляет все положительные примеры, особенно на тестовой выборке. Снижение recall на тесте около 15% - указывает на то, что часть положительных примеров пропускается

Анализ по F1-score: Данная метрика ниже на тестовой выборке, но тк это среднее из precision и recall, значения которых нас вполне устраивают, на данное снижение можно не сильно опираться