機械学習講習会

第六回：部内コンペキックオフ

2025/07/10

@Kobakos32

前回までの復習

1. **ニューラルネットワークの基本構造**：線形層と活性化関数の組み合わせ
2. **誤差逆伝播法**：効率的な勾配計算手法
3. **勾配降下法**：パラメータ最適化の基本手法
4. **PyTorchによる実装**：MNISTでの手書き数字認識

今回は理論の実践応用！
学んだ知識を使って実際のコンペティションに挑戦します。

本日のゴール

1. **Kaggleアカウント**の作成・登録
2. **コンペティション概要**の理解
3. **データセットとタスク**の把握
4. **技術的概念**の理解（埋め込みベクトル、公開/非公開テスト）
5. **ルールと注意事項**の確認
6. **スターターノートブック**の実行
7. **初回提出（ファーストサブ）** の完了

目標：全員がファーストサブを完了する！

もくじ

1. 部内コンペティション概要

今回のコンペティション

テーマ：皮膚疾患の画像分類
データセット：SCIN (Skin Condition Image Network)
期間：1週間（7/10 ～ 7/17）
形式：チーム戦

Home > Blog >

SCIN: A new resource for representative dermatology images

March 19, 2024 · Pooja Rao, Research Scientist, Google Research

age_group	-
sex_at_birth	FEMALE
race_ethnicity_*	BLACK_OR_AFRICAN_AMERICAN
self_fitzpatrick_skin_type	6
textures_*	['FLAT']
body_parts_*	['ARM']
condition_symptoms_*	['ITCHING']
other_symptoms_*	['NO_RELEVANT_SYMPTOMS']
related_category	RASH
condition_duration	7.0 - 28.0 days
dermatologist_gradable_for_skin_condition	YES
condition_labels	eczema, lichen striatus
label_confidence	4, 3
dermatologist_gradable_for_eFST	YES
dermatologist_eFST	5
eMST labels	7, 6

賞品

- **優勝チーム**：図書券 5,000円 * チームメートの数（以下同様）
- **準優勝チーム**：図書券 3,000円 * チームメートの数
- **三位チーム**：図書券 2,000円 * チームメートの数

コンペティションの特徴

実際の医療AIに近い問題設定：

- 皮膚の画像から8種類の疾患を分類
- 画像データ + 患者の症状・属性データを組み合わせ
- 実世界の不均衡データの扱い

予測対象の8つの疾患

1. **Eczema（湿疹）** - 47.6%
2. **Allergic Contact Dermatitis（アレルギー性接触皮膚炎）** - 39.3%
3. **Insect Bite（虫刺され）** - 18.2%
4. **Urticaria（蕁麻疹）** - 14.9%
5. **Psoriasis（乾癬）** - 13.9%
6. **Folliculitis（毛嚢炎）** - 12.1%
7. **Tinea（白癬）** - 9.3%
8. **Impetigo（とびひ）** - 5.5%

特徴：

- **マルチモーダル**：画像データとテーブルデータの組み合わせ
- **多ラベル分類**：1つの画像に複数の疾患が該当する可能性
- **クラス不均衡**：湿疹が最も多く、とびひが最も少ない
- **評価指標**：マクロ平均AUC（各疾患のAUCの平均）

2. Kaggleへの登録

Kaggleとは？

Kaggle：世界最大の機械学習コンペティションプラットフォーム

主な機能：

- **コンペティション**：企業や研究機関が主催する機械学習コンテスト
- **データセット**：公開されている様々なデータセット
- **ノートブック**：ブラウザ上でコードを実行できる環境
- **コミュニティ**：世界中のデータサイエンティストと交流

今回の用途：部内コンペのプラットフォームとして利用

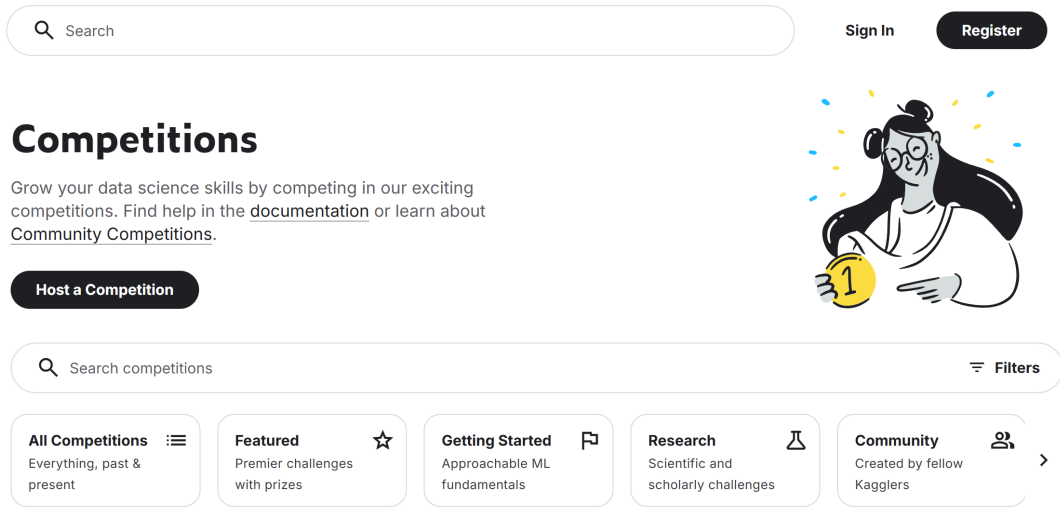
Kaggleアカウントの作成

手順：

1. kaggle.com にアクセス
2. 「Register」をクリック
3. いろいろやる

注意点：

- ユーザー名は後から変更困難
- プロフィールは適切に設定
- 利用規約を確認



右上の黒い「Register」ボタンをクリック

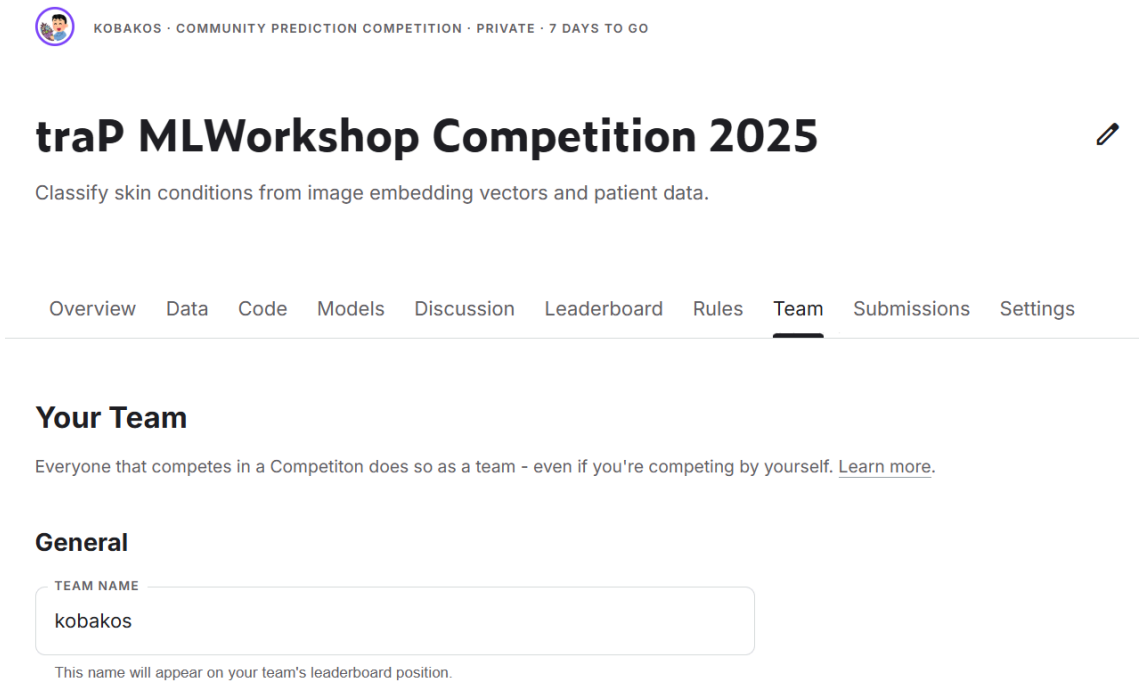
コンペティションへの参加

参加手順：

1. 配布されたコンペティションURLにアクセス
2. 「Join Competition」をクリック
3. ルールに同意
4. データセットをダウンロード

確認事項：

- チーム設定（
- 提出制限の確認
- リーダーボードの確認



チームタブからチームの設定もしてください

3. データセットとタスクの説明

データセット概要

データ構成：

- **訓練データ**：1,333症例（ラベル付き）
- **テストデータ**：898症例（予測対象）
- **画像**：皮膚疾患の写真（埋め込みベクトルに変換済み）
- **テーブルデータ**：45種類の特徴量

特徴量の種類：

- 人口統計（年齢、性別、肌タイプ）
- 症状（質感、部位、症状）

データの詳細

画像データ：

- 症例ごとに1〜3枚の画像
- 特徴ベクトル（6,144次元）に変換済み

テーブルデータ：

- 人口統計：age_group, sex_at_birth, fitzpatrick_skin_type
- 症状：textures_, *body_parts_*, condition_symptoms_*
- その他：撮影条件、期間など（45個の特徴量）

ラベルデータ：

- 8つの疾患それぞれについて0/1のバイナリ
- 複数の疾患が同時に1の場合あり
- 信頼度スコアも含む（高度な手法で利用可能）

評価指標：

- マクロ平均AUC
- 各疾患のAUCを計算し、その平均

4. 技術概念の理解

埋め込みベクトル (Embedding Vectors)

埋め込みベクトルとは？

- 画像や文章などの複雑なデータを数値ベクトルで表現する技術
- 高次元空間で意味的に類似したデータが近くに配置される

今回の場合：

- 皮膚画像 → 6,144次元のベクトル
- 事前訓練済みモデルを使用
- 画像の視覚的特徴を数値で表現

いい感じに特徴が抽出されたベクトルと考えてください

なぜ埋め込みベクトルを使うのか？

生の画像データの問題：

- 高解像度画像は数百万ピクセル
- 計算量が膨大
- ノイズや不要な情報も含む

埋め込みベクトルの利点：

- 重要な特徴のみを抽出
- 次元数が大幅に削減
- 事前訓練の恩恵を受けられる

具体例：

- 元画像：224×224×3 = 150,528次元
- 埋め込み：6,144次元（約25分の1）

公開・非公開テストデータセット

なぜテストデータを分割するのか？

公開テストセット（Public Test）：

- コンペ期間中に結果が見える
- リーダーボードに反映
- 約30%程度のテストデータ

目的：

- 進捗の確認
- 手法の大まかな評価
- モチベーションの維持

非公開テストセット（Private Test）：

- コンペ終了後に結果が判明
- 最終順位の決定
- 約70%程度のテストデータ

目的：

- 真の汎化性能の測定
- 過学習の防止
- 公正な評価

重要：公開テストで高得点でも、非公開で低得点の可能性あり

リーダーボードの読み方

表示される情報：

- 順位
- チーム名
- スコア（マクロ平均AUC）
- 提出時刻

注意点：

- 公開スコアは暫定的
- 最終結果は非公開テストで決定
- 公開テストへの過学習を避ける
- 安定した手法の開発が重要

#	Team	Members	Score	Entries
1	D.Imanishi		0.85	64
2	DeadKey		0.85	197
3	Muku		0.85	91
4	ishikei		0.85	15
5	Optimo		0.85	16
6	Takoi		0.85	24
7	Ogurtsov		0.85	69
8	Devin Anzelmo		0.85	41

5. コンペティションルール

最重要ルール：公開禁止

絶対に守ること：プライベートな譲歩のシェアは禁止

禁止事項：

- コードのGitHub等での公開
- データセットの外部共有
- 解法のSNS投稿
- ブログでの詳細な手法公開

情報共有はディスカッションで！！

その他の注意事項

提出に関して：

- 1日最大50回まで提出可能
- エラーも提出にカウントされます
- ファイルサイズ・形式の確認

技術的制約：

- 外部データの使用禁止

期間・タイムライン：

- コンペ期間：1週間　(2025/07/17 23:59:59 JSTまで)
- 最終提出期限の厳守

6. スターターノートブック解説

スターターノートブックの概要

提供内容：

- 完全に動作するベースラインモデル
- データの読み込みから提出まで一連の流れ
- 2つのヘッドを持つニューラルネットワーク（45個のテーブル特徴量 + 6,144次元画像埋め込み）
- マクロ平均AUC: 約84.8%

学習目標：

- コンペの全体的な流れの理解
- PyTorchを使った実装の経験
- 医療データの特徴の把握

データの読み込みと前処理

```
# データの読み込み
train_cases = pd.read_csv('input/train_cases.csv')
train_labels = pd.read_csv('input/train_labels.csv')
test_cases = pd.read_csv('input/test_cases.csv')
embeddings = torch.load('input/embeddings.pt')

# データの結合
train_data = pd.merge(train_cases, train_labels, on='case_id')

# 特徴量の標準化
scaler = StandardScaler()
train_data_scaled[feature_columns] = scaler.fit_transform(
    train_data[feature_columns]
)
```

ポイント

- 複数ファイルの適切な結合
- 欠損値の処理（0で埋める）
- 特徴量の標準化（45個の特徴量）

データセットクラスの実装

```
class SkinDataset(Dataset):
    def __init__(self, tabular_data, embeddings_dict, labels=None):
        self.tabular_data = tabular_data
        self.embeddings_dict = embeddings_dict
        self.labels = labels

    def __getitem__(self, idx):
        case_id = self.case_ids[idx]

        # テーブル特徴量
        tabular_features = torch.FloatTensor(
            self.tabular_data.iloc[idx][feature_columns].values
        )

        # 画像埋め込み（複数画像の平均）
        image_embedding = torch.mean(
            self.embeddings_dict[case_id], dim=0
        )

        return tabular_features, image_embedding, labels

    def __len__(self):
        return len(self.tabular_data)
```

2つのヘッドを持つモデル

```
class DualHeadModel(nn.Module):
    def __init__(self, tabular_dim, embedding_dim, hidden_dim=256):
        super().__init__()

        # テーブル特徴量用のヘッド
        self.tabular_head = nn.Sequential(
            nn.Linear(tabular_dim, hidden_dim),
            nn.ReLU(),
            nn.Dropout(0.3),
        )

        # 画像埋め込み用のヘッド
        self.image_head = nn.Sequential(
            nn.Linear(embedding_dim, hidden_dim),
            nn.ReLU(),
            nn.Dropout(0.3),
        )

        # 結合後の出力層
        self.output_layer = nn.Sequential(
            nn.Linear(hidden_dim * 2, num_classes),
        )
```

二つの入力をとって、それを間で結合して線形層に通すモデル

モデルの特徴

アーキテクチャの工夫：

- **2つのヘッド**：テーブルデータと画像データを別々に処理
- **特徴融合**：後段で結合して総合的な判断
- **ドロップアウト**：過学習の防止

なぜこの構造なのか？：

- テーブルと画像では情報の性質が異なる
- 解釈性の向上

訓練・評価ループ

```
# 訓練ループ
for epoch in range(num_epochs):
    # 訓練
    train_loss, train_preds, train_labels = train_epoch(
        model, train_loader, criterion, optimizer, device
    )

    # 検証
    val_loss, val_preds, val_labels = validate(
        model, val_loader, criterion, device
    )

    # マクロAUCの計算
    train_auc_mean = calculate_macro_auc(train_labels, train_preds)
    val_auc_mean = calculate_macro_auc(val_labels, val_preds)
```

重要な設定：

- **損失関数**：BCEWithLogitsLoss（多ラベル分類用）
- **評価指標**：マクロ平均AUC
- **早期停止**：ベストモデルの保存

提出ファイルの作成

```
# テストデータでの予測
test_predictions = []
with torch.no_grad():
    for tabular, image in test_loader:
        outputs = model(tabular, image)
        preds = torch.sigmoid(outputs).cpu().numpy()
        test_predictions.append(preds)

# 提出ファイルの作成
submission = pd.DataFrame({'case_id': test_cases['case_id']})
for i, disease in enumerate(target_columns):
    submission[disease] = test_predictions[:, i]

submission.to_csv('submission.csv', index=False)
```

重要な点：

- シグモイド関数で確率に変換
- 各疾患の確率を0-1の範囲で出力
- CSVファイル形式での保存

7. 一般的なワークフローとヒント

一般的なワークフロー

1. **データの理解：**
    - データセットの構造と内容を把握
    - 特徴量の意味を理解
2. **ベースラインの構築**
    - スターターノートブックを実行
    - 初回提出（ファーストサブ）

3 と 4 を繰り返し行って精度を向上させる

3. **モデルの改善：**
    - ハイパーパラメータの調整
    - 特徴量エンジニアリング
4. **評価と検証：**
    - クロスバリデーションの実施
    - リーダーボードでの評価

データ探索のポイント

テーブルデータ：

- 各特徴量の分布
- 疾患との相関関係
- 欠損パターンの分析

画像データ：

- 埋め込みベクトルの分布
- 疾患ごとの特徴の違い
- 次元削減による可視化

ラベル：

- 疾患の共起パターン
- クラス不均衡の確認
- 信頼度スコアの活用

改善アイデア

モデル改善：

- より深いネットワーク
- 異なる活性化関数
- バッチ正規化の追加
- 注意機構の導入

特徴量エンジニアリング：

- 新しい特徴量の作成
- 特徴量選択
- カテゴリカル変数の扱い

学習戦略：

- クラス重みの調整
- 異なる最適化器
- 学習率スケジューリング

アンサンブル：

- 複数モデルの組み合わせ
- 異なる手法の融合

時間管理のコツ

1週間という短期間での戦略：

Day 1-2：理解とベースライン

- データの理解
- スターターコードの実行
- 初回提出

Day 3-5：集中的な改善

- 小さな変更の積み重ね
- 定期的な提出

Day 6-7：最終調整

- 最も効果的な手法の選択
- アンサンブルの検討
- 最終提出の決定

よくある罠と対策

よくある失敗：

- 公開テストセットへの過学習
- 複雑すぎるモデルの構築
- モデルの評価が正しく行えていない
- 時間配分の失敗

対策：

- 堅実なクロスバリデーション
- シンプルなベースラインから開始
- 定期的な振り返り
- 余裕を持ったスケジュール

まとめ

今日やったこと：

1. コンペティション概要の理解
2. Kaggleアカウントの作成
3. 技術概念（埋め込み、テスト分割）の学習
4. 重要ルール（公開禁止）の確認
5. スターターノートブックの解説
6. 改善戦略の検討

次のステップ：

- スターターノートブックの実行
- 初回提出の完了
- チーム内での役割分担
- 改善施策の計画

次回予告

第七回：部内コンペ振り返り（7/18）

- コンペ結果の発表
- 上位チームの手法紹介
- 学んだことの整理
- 今後の機械学習学習について

それまでに：

- コンペへの積極的な参加
- チームでの協力
- 多くの実験と学習

頑張って！良いコンペになることを期待しています