Engineer

機械学習

データの例から課題の解き方を学ぶソフトウェアを作ること

どのようなものか

機械学習は、プログラマーが規則を書き下す代わりに、例から課題の解き方を学ぶソフトウェアを作ることです。答えの分かる事例を数多く与えると、入力と答えの関係を見つけ、その関係から、新しい事例の答えを予測します。名前は、今それを使うソフトウェアのほぼすべてより古く、A. L. Samuelが、IBM Journal of Research and Developmentの1959年7月号に掲載された、チェッカーというゲームの研究の題名に使いました。

用語は、オープンソースのPythonライブラリーscikit-learnに沿って説明すると明確になります。この記事も、その文書に従います。例は表にまとめます。各行が、請求書、顧客、センサーの読み取りなどのサンプルで、各列が、それを説明する数値である特徴量です。学ばせたい答えは、目的変数です。scikit-learnでは、学ぶものを推定器と呼びます。どの推定器にも、見せた行から学習する `fit` メソッドがあります。予測を行う推定器は、 `predict` で新しい行の答えを返し、変換器は、 `transform` でサンプルの表を変換したものを出力します。

大半の仕事は、二つに分かれます。教師あり学習では、学習する各行に目的変数があります。期日どおりか遅れたかのようなカテゴリなら、分類問題です。支払いまでの日数のような数量なら、回帰問題です。教師なし学習には、目的変数がなく、似た行動の顧客をまとめるなど、ソフトウェアが自ら構造を探します。scikit-learnの文書は、それぞれに章を設け、続いてモデル選択と評価を扱います。実務上の難しさの大半は、そこにあります。

少なくともこの意味での機械学習は、賢そうに見えるソフトウェアすべてを指す言葉ではありません。一定額を超える請求書に印を付ける規則は、人が書き、その指示どおりに動きます。請求書に印を付けるモデルは、過去の請求書から規則を導いたので、誰も書き下していない形で間違え、学習するデータが変わると動作も変わります。

概念を判断に生かす

「機械学習とは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 使用中のシステムに接続し、自社の専門家が用意した事例でテストし、自社の担当者が監督できる、特定のモデルに縛られないエージェントをつくります。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 268検査を持つリポジトリの検査基盤を構築 — 268 コミット検査と、検査自体の146テスト

268検査を持つリポジトリの検査基盤を構築 お問い合わせ →

なぜ重要なのか

仕事によっては、説明するより例を示す方がずっと簡単です。どの請求書の支払いが遅れるかを完全な規則にできる人はいませんが、大半の企業には、結果を記した何年分もの請求書があります。機械学習は、その記録を動く規則に変え、同じような多数の問いを同時に扱えます。研究から、日常の業務ソフトウェアに広がった理由です。

代価は、それまでコードが担っていた仕事を、データが担うことです。履歴に誤り、重複、抜けがあれば、モデルはそれも学びますが、その事実はコードには現れません。D. SculleyとGoogleの同僚九人は、NIPS 2015の会議録の論文*Hidden Technical Debt in Machine Learning Systems*で、その後に起きることを説明しました。最初の図は、はるかに大きいシステムの中央に、学習コードを小さな黒い箱として描き、説明文は周囲の基盤を巨大で複雑と呼びます。一節では、成熟したシステムは、機械学習コードが多くても5%、汎用パッケージにデータを出し入れする連携用コードが少なくとも95%になる可能性があると見積もります。

データへの依存は、コードへの依存より高くつくとも論じます。コンパイラーとビルドシステムは、どのコードがどれに依存するかを追いますが、データについて同じことをする道具は、はるかに少ないため、列の意味が変わっても気づかれない場合があります。また、「Changing Anything Changes Everything」、何かを変えるとすべてが変わる、というCACEの原則を名付けます。一つの入力の分布を変えると、ほかのすべての入力をモデルが使う方法も、変わる可能性があります。

したがって、モデルの作成は安くても、維持には費用がかかります。役立つかどうかを決める仕事は、主にデータです。出所、各顧客が一回現れるか三回現れるか、結果の記録が正しいか、上流のシステムが違うものを送り始めたら誰が気づくかです。

実際の活用例

請求書を発行した日に、どの支払いが遅れそうかを知り、期日後ではなく期日前に顧客に電話したい企業を考えます。書き出したデータには、過去の請求書一件につき一行があり、発行日順に並びます。発行日に分かっていた四つの列は、金額、認めた支払い猶予の日数、その前の一年に顧客が受け取った請求書の数、そのうち遅れた数です。五つ目の列が結果で、支払いが遅れたら1、期日どおりなら0です。

次のスクリプトは、この記事用に作り、2026年10月に、scikit-learn 1.9.1とNumPy 2.5の文書と一行ずつ照合しました。古い請求書でモデルを学習させ、新しい方の四分の一で試します。

```
# invoices.csv: 請求書一件につき一行。発行日順に並べ、数値だけを記す。
# 最後の列は、支払いが遅れたら1、期日どおりなら0。
import numpy as np
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import balanced_accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

data = np.loadtxt("invoices.csv", delimiter=",", skiprows=1)
X, y = data[:, :-1], data[:, -1].astype(int)

# 新しい方の四分の一を取り分け、学習には使わない。
X_old, X_new, y_old, y_new = train_test_split(
    X, y, test_size=0.25, shuffle=False
)

model = make_pipeline(StandardScaler(), LogisticRegression())
baseline = DummyClassifier(strategy="most_frequent")
for name, candidate in [("model", model), ("baseline", baseline)]:
    candidate.fit(X_old, y_old)
    score = balanced_accuracy_score(y_new, candidate.predict(X_new))
    print(name, round(score, 3))
```

アルゴリズムの選択以上に重要な判断が、四つあります。

学習前に、新しい四分の一を取り分けます。 `shuffle=False` により、`train_test_split` は行の順を保ち、最後の四分の一をテスト集合にします。モデルは古い請求書で学び、新しいもので評価され、実際の使い方と一致します。既

定の `shuffle=True` なら分割は無作為になり、学んだ一部の請求書より古い請求書でも試すことになります。

スケーリングをパイプライン内に置きます。 `StandardScaler` は、各特徴量から平均を引き、分散が1になるように調整します。 `make_pipeline` が分類器につなぐため、平均と分散は学習行だけから求まり、新しい四分の一の情報はモデルに入りません。

評価には、均衡精度を使います。 二つのクラスについて、それぞれを正しく当てた割合、つまり遅れた請求書を検出した割合と、期日どおりのものを認識した割合を平均します。十件に一件が遅れるなら、通常の正解率では、毎回「期日どおり」と答えるモデルが高く評価されます。均衡精度では、そうなりません。

モデルは、基準を上回る必要があります。 `DummyClassifier` に `strategy="most_frequent"` を指定すると、特徴量を無視し、常に最も多い答えを返します。二クラスの均衡精度は0.5です。期日どおりの請求書はすべて当たり、遅れるものはすべて外れるためです。これを明確に上回らないモデルは、通常の正解率がどれほど高くても、有用なことを学んでいません。

どれも、誤った行からモデルを救ってはくれません。一人の顧客が三つの記録として存在すれば、遅れた請求書の数が三つに分かれ、実際にはなかった履歴から学びます。AIデータ準備のページが扱うのは、その土台です。顧客ごとの単一の記録、各人が見るべきものだけを見られる権限、どの答えも根拠のデータまで追えることです。モデルでは、最後の性質が、奇妙な予測から、それを生んだ行まで遡ることを可能にします。

問題が起きるところ

未来の情報が学習に入ります。 scikit-learnのよくある落とし穴のガイドは、予測時には存在しない情報でモデルを作ることを、データリークと呼びます。請求書の例で足したくなる列は、送った督促の数です。督促は支払期限を過ぎてから送るため、遅れを非常によく当てます。分割前に、全行でスケーラーや欠損値の補完器を調整しても、目立たない形で同じ情報の混入が起きます。ガイドの規則は、テストデータに対して決して `fit` を呼ばないことです。各列の値がいつ分かるかを書き、先に分割し、前処理をすべてパイプライン内に置いて、学習行だけで調整します。

既に見たものでモデルを評価します。 学習した目的変数を繰り返すだけのモデルは、自分の学習行では満点でも、新しいものには役立ちません。交差検証のガイドは、これを過学習と呼びます。さらに分かりにくいのは、多数の設定を試して、取り分けた行で最もよかったものを残す場合です。その行についての知識が、選択に漏れ込みます。設定は学習部分の交差検証で比べます。行を複数の分割片に分け、一つずつテスト集合を担当させます。取り分けた行を見るのは、最後の一回だけにします。

常に同じ答えでも出せる高い正解率です。 一方のクラスが少なければ、多い方が正解率を支配します。ガイドは、不正検知のように少ないクラスがあると、分割片にそのクラスの例が一つも入らない場合があると注意します。均衡精度などクラス別の指標を報告し、`DummyClassifier` と比べ、各分割片でクラスの比率をほぼ等しくする層化分割を使います。分類器に対して分割数を渡した場合、`cross_val_score` は既定でこれを使います。

時間で変わるデータを無作為に分けます。 交差検証のガイドは、サンプルが独立で同一の分布に従うという仮定は実際にはほとんど成り立たず、時系列順のデータは、学習したものより後の観測で試すべきだと述べます。無作為な分割は、モデルに未来をのぞかせ、実際には届かない成績を報告します。例のように日付で分け、交差検証には `TimeSeriesSplit` など時間を考慮する分割器を使います。ガイドによれば、各分割片は同じ長さの時間範囲を覆う必要があります。

学習と本番で前処理が違います。 スケーリングした特徴量から学んだモデルに、本番が生の値を送ると、エラーのないまま予測が間違います。落とし穴のガイドは、学習に使った変換を、テストデータにも本番データにも同じように適用するよう求めます。モデルに、事前の加工の説明書を添えるのではなく、学習済みのパイプラインを一つのオブジェクトとして配布します。

公開後に世界が変わります。 昨年の請求書から学んだモデルは、顧客が今も同じ行動をすると仮定します。Sculleyらは、予測の偏りを追うよう勧めます。意図どおりに動くシステムでは、通常、予測した各ラベルの割合は、実際に観測した割合と一致するはずで、差が開くことが、世界が変わった最初の兆候になる場合がよくあります。比較を続け、最近のデータで再学習し、別のチームが管理する入力信号は、固定し、版を付けたコピーを保持します。上流の善意の変更が、気づかれないままモデルを変えないようにするためです。

参考資料

- ユーザーガイド — 教師あり学習、教師なし学習、モデル選択と評価についての、章ごとのscikit-learnの参照資料です。

- はじめに — 推定器のインターフェースを、学習と予測、変換器、パイプライン、交差検証の短い例で示します。
- よくある落とし穴と推奨する実践 — 不統一な前処理、データリーク、無作為性について、誤ったコードと正しいコードを並べます。
- 交差検証：推定器の性能評価 — 学習データで試すと誤解する理由と、グループ化、不均衡、時系列のデータに適した分割です。
- 機械学習システムの隠れた技術的負債 — GoogleのSculleyらによるNIPS 2015の論文です。モデル自体を超える、本番運用の費用を説明します。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「機械学習とは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/machine-learning/