仕事ではじめる機械学習
4章 システムに機械学習を組み込む

M2 倉地亮介

目次

システムに機械学習を含める流れ

1. 問題を定式化する
2. 機械学習をしないで良い方法を考える
3. システム設計・誤りをカバーする方法を考える
4. アルゴリズムを選定する
5. 特徴量，教師データとログの設計をする
6. 前処理をする
7. 学習・パラメータチューニング
8. システムに組み込む

システム設計

- 分類や回帰などの教師あり学習の場合，学習と予測の2つのフェーズがある
- 学習フェーズでは2種類のタイミングがある
  - バッチ処理での学習
  - リアルタイム処理での学習

混乱しやすい「バッチ処理」と「バッチ学習」

- バッチ処理
  - 一括で何かを処理すること，またその処理そのものを指す
- リアルタイム処理
  - 刻々と流れてくるデータに対して逐次処理をすること

※バッチ学習を**一括学習**，オンライン学習を**逐次学習**と表現する

一括学習と逐次学習

- モデル学習時のデータの保持の仕方が異なる
  - 一括学習
    - 重み計算のためにすべての教師データを必要とし，全データを用いて最適な重みを計算
    - データが増えるとその分メモリ量も増加する
  - 逐次学習
    - 教師データを1つ与えて、その都度重みを計算
    - メモリに保持されるデータは，その時のデータと計算された重みだけ

学習時の必要とするデータの塊が違い，学習時の最適化の方針が違うだけ

取りうる処理と学習の組み合わせ

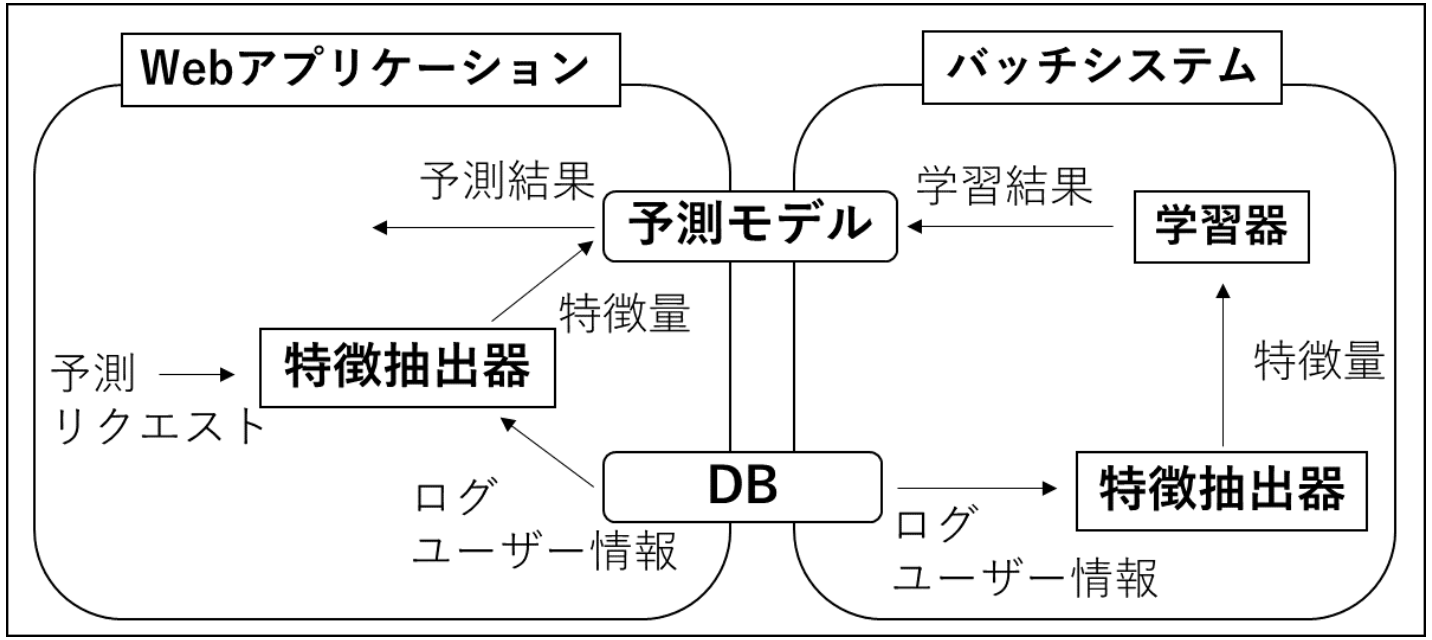
1. バッチ処理で一括学習
2. バッチ処理で逐次学習
3. リアルタイム処理で一括学習
4. リアルタイム処理で逐次学習

- 2.の「バッチ処理で逐次学習」とはどのようなものなのか？
  - バッチ処理でまとまったデータを一括処理をするけれど，最適化方針は逐次学習すること

バッチ処理で学習を行う3つの予測パターン

1. バッチ処理で学習＋予測結果をWebアプリケーションで直接算出する(リアルタイム処理で予測)
2. バッチ処理で学習＋予測結果をAPI経由で利用する(リアルタイム処理で予測)
3. バッチ処理で学習＋予測結果をDB経由で利用する（バッチ処理で予測）

バッチ処理で学習＋予測結果をWebアプリケーションで直接算出する(リアルタイム処理で予測)



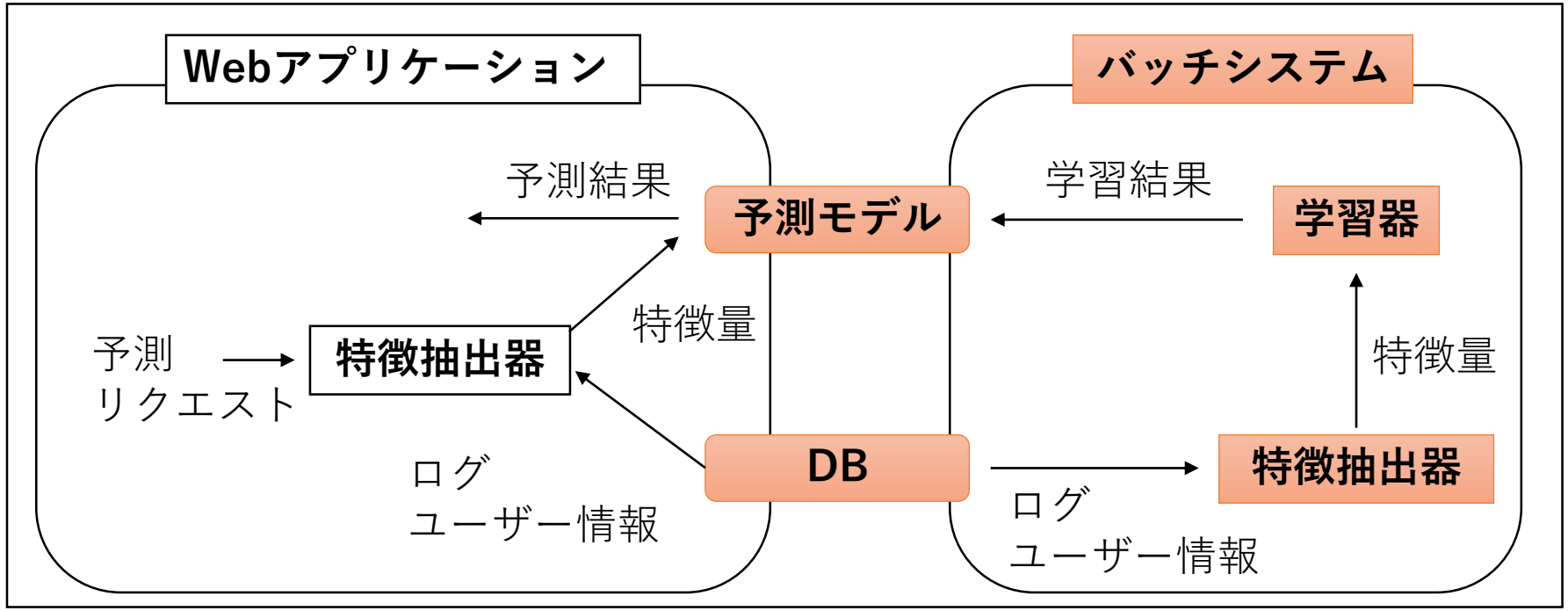
- 最も素朴な方法
  1. バッチ処理で一括学習
  2. 予測モデルをリアルタイム処理で利用
- 特徴
  - 予測はリアルタイム処理が必要
  - Webアプリケーションとバッチシステムの言語が同一

比較的単純な構成のため試すのも容易で，小規模で試してみるのに適したパターン

バッチ処理で学習＋予測結果をWebアプリケーションで直接算出する(リアルタイム処理で予測)

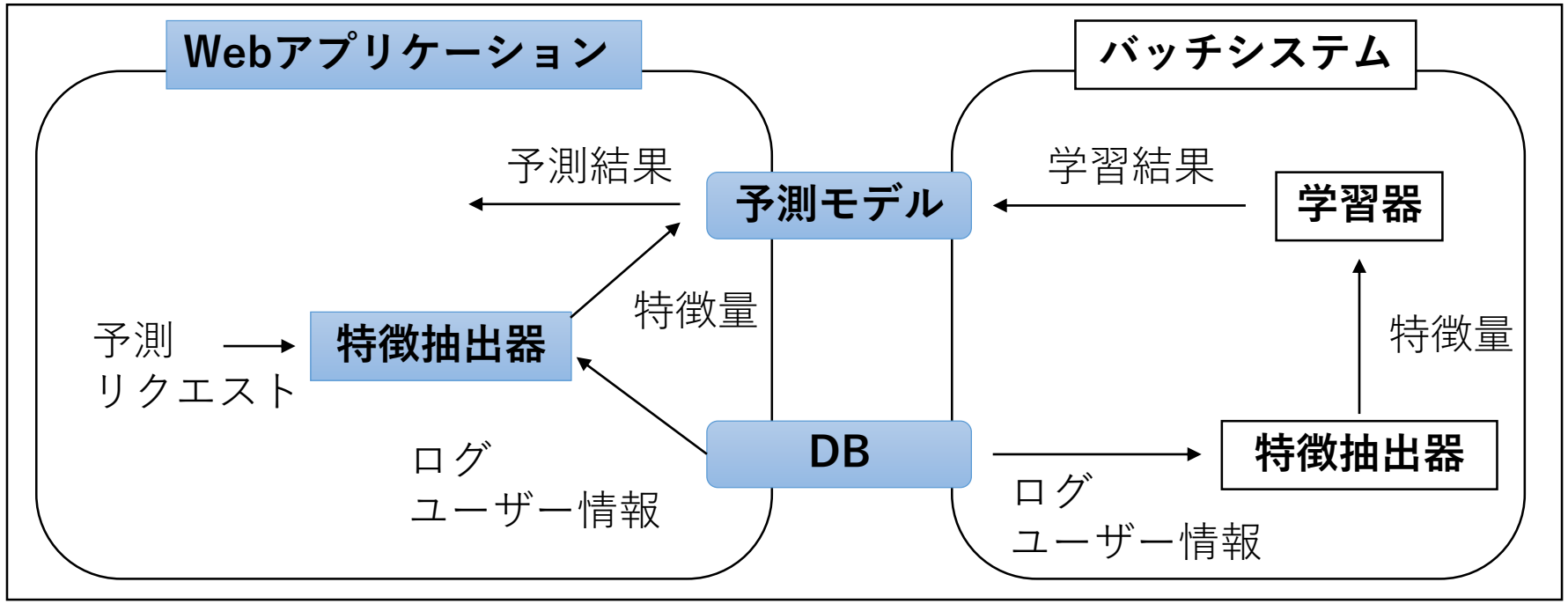
- このパターンが用いられる場面
  - 入力データが事前に用意できず，予測の結果を低遅延で使いたい場合
    - e.g. 広告配信の最適化
- 遅延を抑えるために
  - データのフェッチ，前処理，特徴抽出，予測といった一連の処理が低遅延で完結することが望ましい
    - データや特徴量を前処理してDBに格納しておく
- 機械学習の処理部分とWebアプリケーションが密結合になりやすい
  - 機械学習のプロトタイプはPythonで行い，そのロジックをWebアプリケーションで利用しているJavaScriptやRubyで実現する

パターン1：学習フェーズ



1. DBから予め蓄積されたログやユーザー情報を取得し，特徴量を抽出する
2. 特徴量をもとに何らかのモデルを学習する
3. 学習結果は学習済みモデルをシリアライズして保存したものをストレージに保存する

パターン1：予測フェーズ

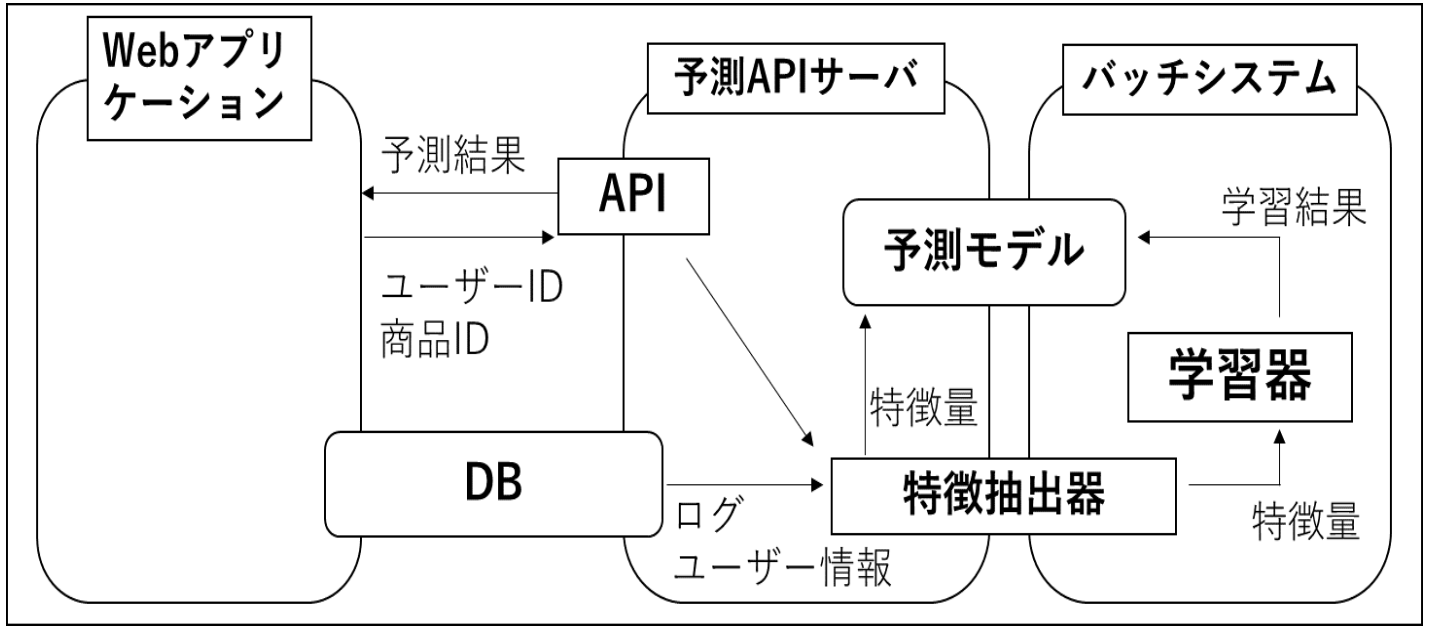


Webアプリケーションが何かしらのイベントをトリガーに予測を要求する

- 例えば，スパムかどうかを判定したいコメントが投稿されたとする

1. イベント発生時に予測をしたい対象の情報をDBから取得し，特徴量を抽出
2. 学習済みモデルを読み込み，抽出した特徴量を入力して予測結果を出力

バッチ処理で学習＋予測結果をAPI経由で利用する(リアルタイム処理で予測)



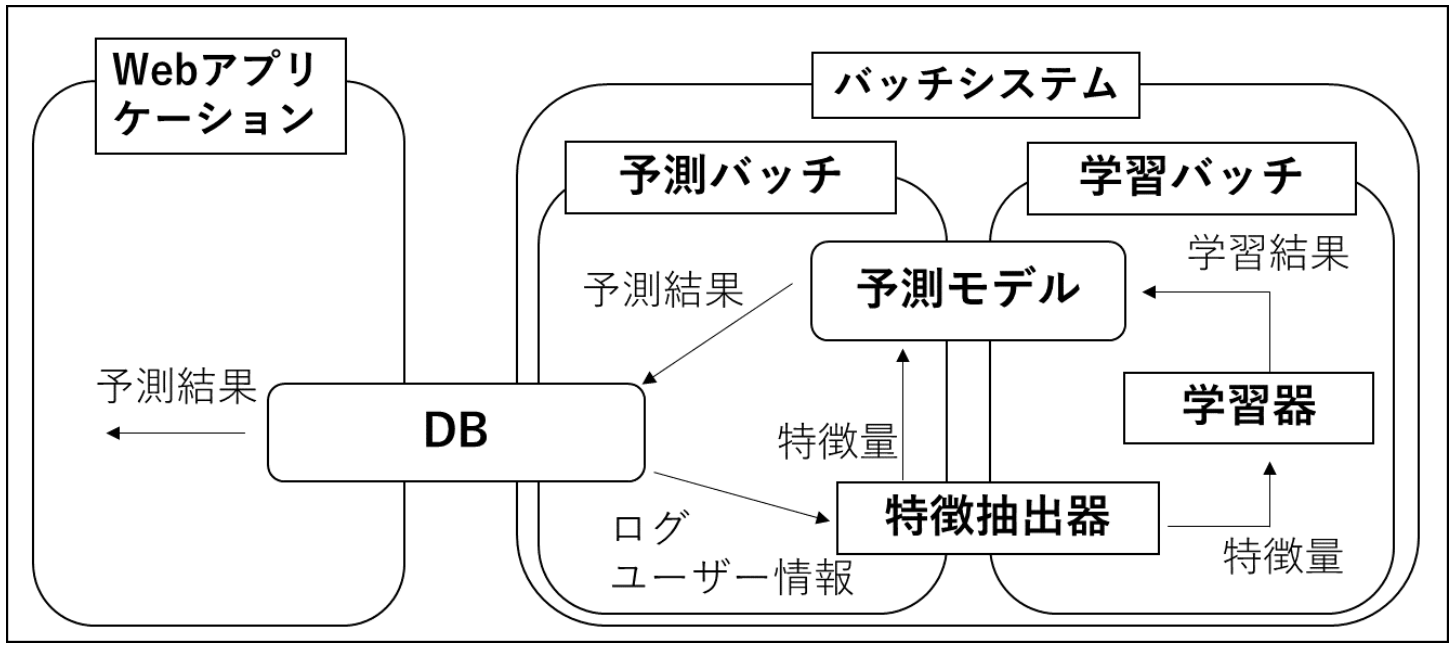
- APIサーバーを用意するパターン
- 特徴
  - Webアプリケーションと機械学習に使う言語を分けられる
  - Webアプリケーション側のイベントに対してリアルタイム処理で予測

予測結果を利用する場合にAPI経由のリアルタイム処理で予測を行うパターン

バッチ処理で学習＋予測結果をAPI経由で利用する（リアルタイム処理で予測）

- メリット
  - Webアプリケーションとの結合が疎になる
    - 複数モデルによるA/Bテストを行う場合に，モデルの比較がしやすい
- デメリット
  - パターン1に比べると遅延が大きくなる
    - APIサーバと予測結果を利用するクライアントの間で通信が発生するから
  - 遅延を小さくするために
    - HTTPやRPCなどのリクエストを投げる部分を非同期にする
    - 予測処理の結果を待つ間に他の処理を並列で進める

バッチ処理で学習＋予測結果をDB経由で利用する(バッチ処理で予測)



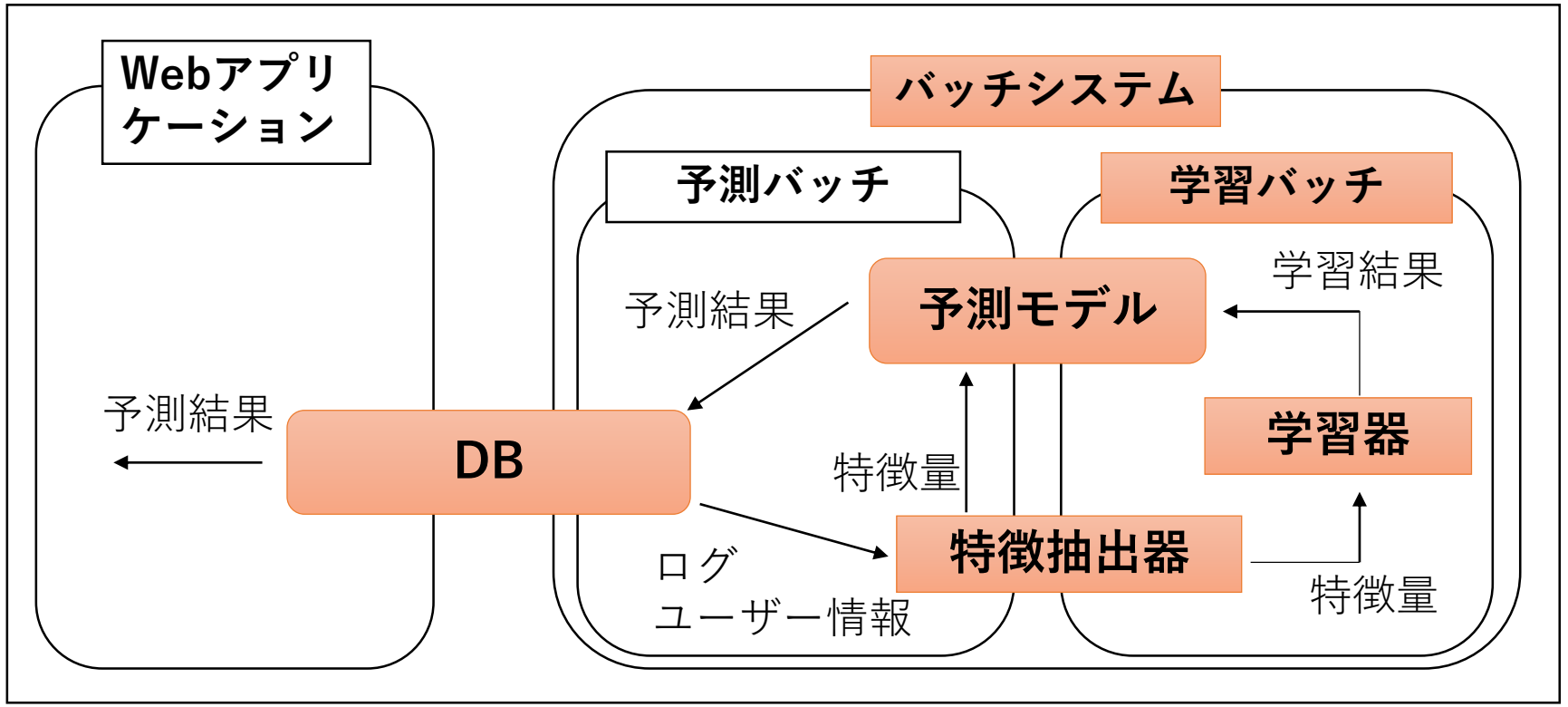
- 使い勝手の良いパターン
  1. 教師あり学習のモデルを一括学習
  2. そのモデルを使った予測をバッチ処理で行う
  3. 予測結果をDBに格納する

- 特徴
  - 予測に必要な情報は予測バッチ実行時に存在する
  - イベントをトリガーとして即時に予測結果を返す必要がない

バッチ処理で学習＋予測結果をDB経由で利用する(バッチ処理で予測)

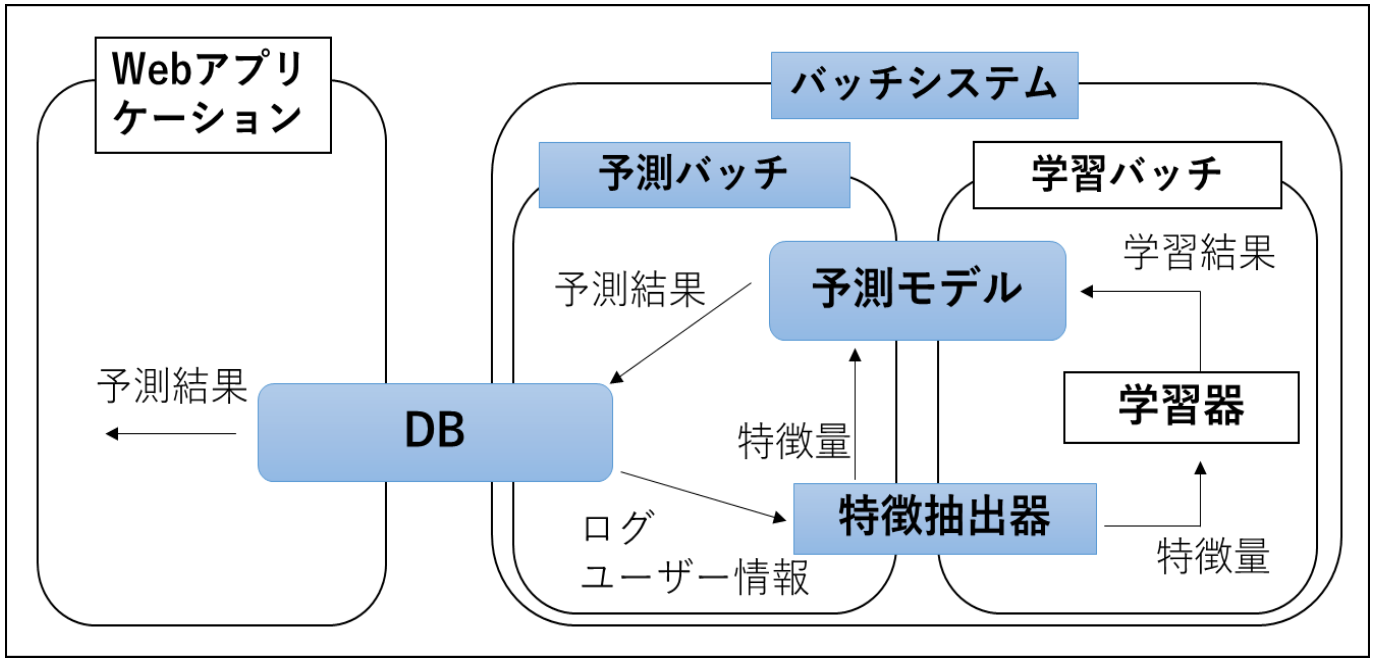
- このパターンが用いられる場面
  - 予測の頻度が頻繁でない対象や結果に向いているパターン
    - e.g. 商品説明などの変化しにくいコンテンツを6時間ごとのバッチで分類
- メリット
  - Webアプリケーションと機械学習の学習・予測を行う言語がそれぞれ異なっていてもよい
  - 予測処理に多少時間がかかる場合でもアプリケーションのレスポンスに影響しない
- デメリット
  - 予測対象となるコンテンツが増えていくと，それに比例して処理時間も増える

パターン3：学習フェーズ



- ログやユーザー情報から特徴を抽出してモデルを一括学習
- 学習済みモデルは，シリアライズしてストレージに保持し予測フェーズで使用する

パターン3：予測フェーズ



- DB中のデータから特徴量を抽出し，バッチ処理で予測する
- 予測結果をWebアプリケーションで利用できる形にしてDBへ格納

- 予測対象となるコンテンツが増えていくと，それに比例して処理時間も増えるので注意が必要
  - 新規で登録された差分のコンテンツだけ予測を行う
  - 並列数を増やして予測処理を行う
  - 分散可能な環境で予測する

各パターンのまとめ

パターン	一括学習＋直接予測	一括学習＋API	一括学習＋DB
予測	リクエスト時	リクエスト時	バッチ
予測結果の提供	プロセス内API経由	REST API経由	共有DB
予測リクエストから結果までのレイテンシ	○	○	◎
新規データ取得から予測結果を渡すまでの時間	短	短	長
一件の予測処理にかけられる時間	短	短	長
Webアプリケーションとの結合度	密	疎	疎
Webアプリケーションのプログラミング言語と	同一	独立	独立

ログ設計

- 機械学習で必要な教師データをどのように取得するかを設計すること
  - Webサーバのアプリケーションログ
  - どこをどうクリックしたかなどのユーザの行動ログ
- ログの特徴
  - DBなどのデータと異なりスキーマがない
  - 記録してないデータを後から改めて取得するのが困難
- 特徴量や教師データに使えそうな情報
  1. ユーザー情報
  2. コンテンツ情報
  3. ユーザーの行動ログ

ログを保持する場所

ユーザーの行動ログはデータ量が多くなる

- 分散RDBに格納する
  - **分散DB**：ネットワーク上に複数存在するDBをあたかも一つのDBであるかように利用する仕組み

- 分散処理基盤Handoopクラスター—HDFSに格納する
  - **Handoop**：構造化されていないデータを、高速に処理出来るプラットフォーム
  - **HDFS**：ファイルを一定のブロックに分割し，複数の記憶装置に分散して保存

- オブジェクトストレージに格納する
  - 各オブジェクトが互いに依存関係のない状態で保存されている
  - データを移動，複製，分散化しやすい

ログを設計する上での注意点

大規模データの転送コスト

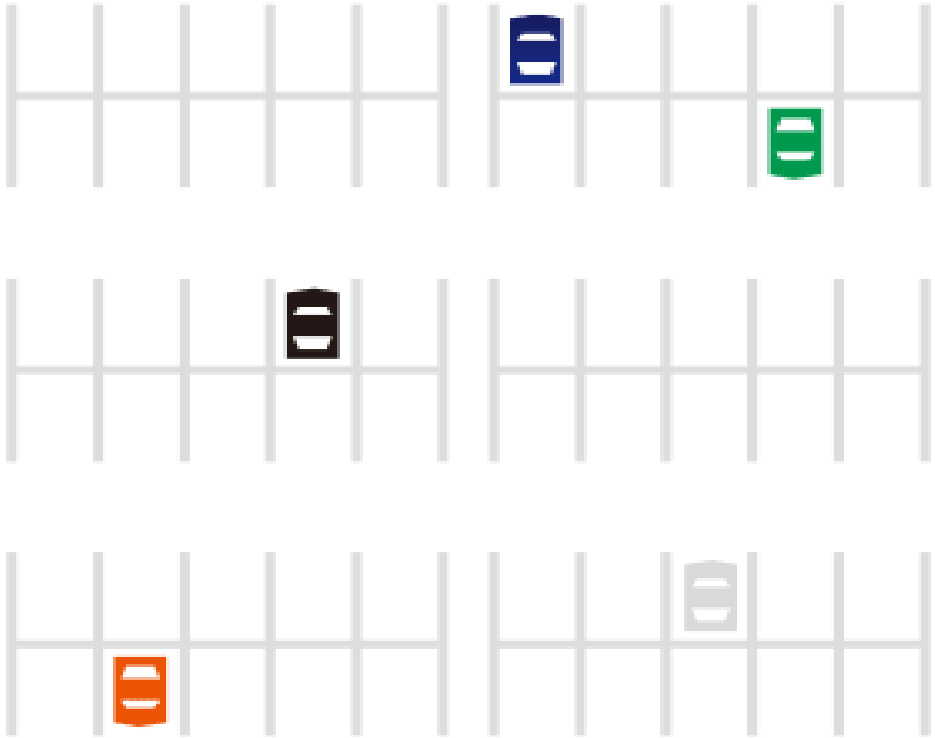
- バッチ処理を行うサーバーにデータを転送する際，時間がかかる
- 時間を抑えるために
  - 分散RDB上でSQLを使って前処理をできるようにするのが望ましい
- 大規模なデータに対して，複雑な前処理を定期的に実行する必要がある場合
  - 出来る限りローカルマシンにダウンロードしない工夫をする
    - e.g. Amazon S3に置いたデータをAmazon EMRで加工する

4.4 まとめ

- システム設計
  - 一括学習をしてから得られたモデルから，予測結果をどのように呼び出すかによって3つのパターンがある
    - バッチ処理で学習＋予測結果をWebアプリケーションで直接算出する
    - バッチ処理で学習＋予測結果をAPI経由で利用する
    - バッチ処理で学習＋予測結果をDB経由で利用する
- ログ設計
  - 特徴量や教師データに使いうる情報
  - データの保持方法と転送方法

P21 オブジェクトストレージ

ファイルストレージは郊外のショッピングモールの駐車場のイメージ。
例えば、「Aブロックの59番」といった具合に、車を停めた場所を覚えておいて、自分で探す必要がある



オブジェクトストレージはタワーパーキングのイメージ。
どこに停めたか覚える必要がなく、IDさえ分かれば車を出してもらえる

P22 データウェアハウス