Tuesday, December 17, 2019

Ch 01: The Machine Learning Landscape

What Is Machine Learning?

- code: https://github.com/ageron/handson-ml2
- first mainstream application - spam filter (Skynet)
- learn from data
- training set -> training instance (each example)

Why Use Machine Learning?

- great for
  - problems for which existing solutions require a lot of hand-tuning or long lists of rules -> ML algorithm simplifies code and performs better
  - complex problems for which there is no solution using traditional approach -> ML techniques can find a solution
  - fluctuating environments -> ML systems can adapt to new data
  - getting insights about complex problems and large amounts of data
- machine learning can help humans learn
  - ML algorithms can be inspected to see what they have learned -> reveals new trends -> better understanding of the problem
- **data mining** = applying ML techniques to dig into large amounts of data can help discover patterns that weren't immediately apparent

Types of Machine Learning Systems

- classification in broad categories
  - trained with human supervision
    - supervised, unsupervised, semisupervised, Reinforcement Learning
  - learn incrementally on the fly
    - online, batch learning

- comparing known data points or detect patterns in the training data and build predictive model
  - instance-based versus model-based learning

SUPERVISED/UNSUPERVISED LEARNING

SUPERVISED LEARNING

- training data includes desired solutions = labels
- used for
  - **classification**
  - **regression**
    - prediction of target numeric values, given set of features = predictors (mileage, brand) and their labels (price)
    - some regression algorithms can be used for classification too (logistic regression)
- supervised learning algorithms
  - k-nearest neighbors, linear regression, logistic regression, support vector machines (SVM), decision trees and random forests, neural networks

UNSUPERVISED LEARNING

- **clustering**
  - K-means, DBSCAN, hierarchical cluster analysis
  - hierarchical clustering - subdividing each group into smaller groups
- **anomaly detection and novelty detection**
  - one-class SVM, isolation forest
  - feed complex, unlabeled data -> output 2D or 3D representation of your data
  - remove outliers before feeding it to another ML algorithm
  - novelty detection = expect to see only normal data during training
  - anomaly detection can perform well even with some outliers during training
- **visualization and dimensionality reduction**

- principal component analysis (PCA), kernel PCA, locally-linear embedding, t-distributed stochastic neighbor embedding
- visualization
  - try to preserve structure (keep separate clusters in input space from overlapping in visualization) - helps you understand how data is organized and identify patterns
- dimensionality reduction
  - goal - simplify data without losing too much information
  - merge several correlated features into one (mileage correlated to age) = **feature extraction**
  - reduce dimension of your training data before feeding it to another ML algorithm

- association rule learning
  - apriori, eclat
  - discover relations between attributes

SEMISUPERVISED LEARNING

- partially labeled data (usually a lot of unlabeled)
- e.g. Google Photos - recognizes the same person is in photos 1,2,5 (clustering), you need to tell who the person is
- combination of unsupervised and supervised algorithms
- deep belief networks (DBNs)
  - based on unsupervised restricted Boltzmann machines (RBMs) and then the whole system is fine-tuned using supervised learning

REINFORCEMENT LEARNING

- learning system (= **agent**) can observe the environment, select and perform actions
- gets rewards or penalties
- learns by itself what the best strategy (= **policy**) is to get the most rewards

BATCH AND ONLINE LEARNING

BATCH LEARNING

- incapable of learning incrementally
- trained -> launched into production
- = **offline learning**
- to train on new data -> train it from scratch

ONLINE LEARNING

- train system incrementally by feeding it data instances sequentially - individually or by small groups (= **mini-batches**)
- advantages
  - need limited computing resources (don't have to save learned data)
  - adapt to change rapidly
- **out-of-core learning** = training systems on huge data sets that can't fit in one computer
  - load part of data -> run training -> repeat until now no data left
- learning rate = how fast it adapts to changing data
  - when quick to adapt, it also quickly forgets the old data
- monitor using anomaly detection algorithm to prevent drop in performance when bad data is fed to system

INSTANCE-BASED VERSUS MODEL-BASED LEARNING

INSTANCE-BASED LEARNING

- system learns examples by heart, then generalizes to new cases by comparing them to learned examples using similarity measure
- e.g. flag all emails that have already been flagged by users or are similar to flagged emails (= **measure of similarity**)

MODEL BASED LEARNING

- build model of set of examples, then use it to make predictions
- utility or fitness function = measures how good model is
- cost function = measures how bad model is

- study data -> select model -> train -> apply model to make predictions

---

Main Challenges of Machine Learning

PROBLEMS WITH DATA

- insufficient quantity of training data
  - unreasonable effectiveness of data
    - one paper showed that very different ML algorithms performed almost identically well on complex problem of natural language disambiguation once they were given enough data
    - data matters more than algorithm
- non-representative training data
  - data needs to be representative of the new cases you want to generalize to
  - **sampling noise** - if dataset is too small or non-representative
  - **sampling bias** - sampling method is flawed
- poor-quality data
  - training data is full of errors, outliers and noise
  - clean the dataset
    - discard or manually fix outliers
    - if some instances are missing a few features (some customers didn't specify age), ignore the attribute, ignore the instances, fill in the missing values, or train one model with the feature and one model without it
- irrelevant features
  - training data must contain enough relevant features and not many irrelevant
  - **feature engineering** - coming up with good set of features to train on, involves
    - feature selection - selecting the most useful features to train on among existing features
    - feature extraction - combing existing features to produce more useful one (dimensionality reduction algorithm)

- creating new features by gathering new data

PROBLEMS WITH ALGORITHMS

- model performs well on training data, but it doesn't generalize well
- model detect patterns in the noise -> the patterns don't generalize well to new data set
- happens when model is too complex relative to the amount and noisiness of thinning data
- solutions
  - simplify model by selecting one with fewer parameters (polynomial -> linear), reducing the number of attributes in training data or constraining the model
  - gather more training data
  - reduce noise in training data (fix data errors, remove outliers)
- **regularization**
  - constraining a model to make it simpler
  - reduces risk of overfitting
  - **degree of freedom**
    - e.g. linear model has slope and height, model that is allowed to modify only slope has one degree of freedom
  - **hyperparameter**
    - controls the amount of regulation to apply during learning
    - parameter of learning algorithm, not of the model
    - set prior to training and remains constant
- under-fitting the training data
  - model is too simple to learn the underlying structure of data
  - solutions
    - select more powerful model with more parameters
    - feeding better features the learning algorithm (feature engineering

  - reducing the constraints on the model (reducing the regularization hyper parameter)

Testing and Validating

- split data into training set and test set
- **generalization error** = out-of-sample error - error rate on new cases
- training error low, generalization error high -> overfitting

Hyper-parameter Tuning and Model Selection

- **holdout validation** - hold out part of training set (= validation set) to evaluate several candidate models and select the best one
  - train multiple models with various hyper-parameters on reduced training set and you select a model that performs best on the validation set
  - after this holdout validation process, you train the best model on the full training set and this gives you the final model
  - evaluate final model on test set to get an estimate of the generalization error
- **cross-validation** - use many small validation sets
  - each model is evaluated once per validation set after it is trained on the rest of the data
  - average out all evaluations of model - mode accurate measure of its performance, but longer training time

Data Mismatch

- validation set and test set must be representative
- **train-dev set**
  - solution to problem - when performance on validation set is not good, you don't know if it is because of overfitting or non-representative data
  - after model is trained, evaluate it on train-dev set
    - performs well -> model isn't overfitting the training set

    - performs poorly -> model overfits training set

- **no free lunch theorem**
  - model = simplified version of observations
  - simplification - discard detail that are unlikely to generalized
    - to decide which data to discard, make assumptions
  - if you make no assumptions, there is no reason to prefer one model over any other
  - there is no model that is guaranteed to work better
    - to find best model, you must evaluate all models
      - not possible -> make assumptions about data -> evaluate only a few reasonable models