Machine Learning

Introduction

Albert Bifet(@abifet)

Data Science

Data Science is an interdisciplinary field focused on extracting knowledge or insights from large volumes of data.

Data Scientist

DATA SCIENTIST
Math
Statistics
Programming
Database
Domain Knowledge
Soft Skills
Communication
Visualization

Figure: `http://www.marketingdistillery.com/2014/11/29/is-data-science-a-buzzword-modern-data-scientist-defined/`

Data Science

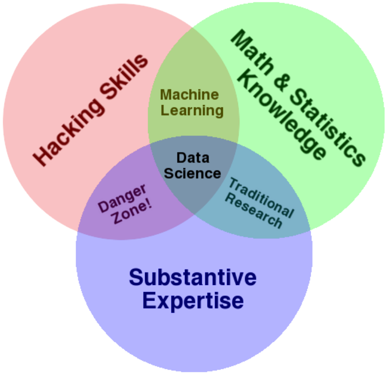


Figure: Drew Convay's Venn diagram

Classification

Definition

Given n_C different classes, a classifier algorithm builds a model that predicts for every unlabelled instance I the class C to which it belongs with accuracy.

Example

A spam filter

Example

Twitter Sentiment analysis: analyze tweets with positive or negative feelings

Classification

Example

Data set that describes e-mail features for deciding if it is spam.

Contains "Money"	Domain type	Has attach.	Time received	spam
yes	com	yes	night	yes
yes	edu	no	night	yes
no	com	yes	night	yes
no	edu	no	day	no
no	com	no	day	no
yes	cat	no	day	yes

Assume we have to classify the following new instance:

Contains "Money"	Domain type	Has attach.	Time received	spam
yes	edu	yes	day	?

k-Nearest Neighbours

k-NN Classifier

- Training: store all instances in memory
- Prediction:
  - Find the *k* nearest instances
  - Output majority class of these *k* instances

Bayes Classifiers

Naıve Bayes

- Based on Bayes Theorem:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$

$$posterior = \frac{prior \times likelikood}{evidence}$$

- Estimates the probability of observing attribute a and the prior probability $P(c)$
- Probability of class c given an instance d:

$$P(c|d) = \frac{P(c)\prod_{a \in d} P(a|c)}{P(d)}$$

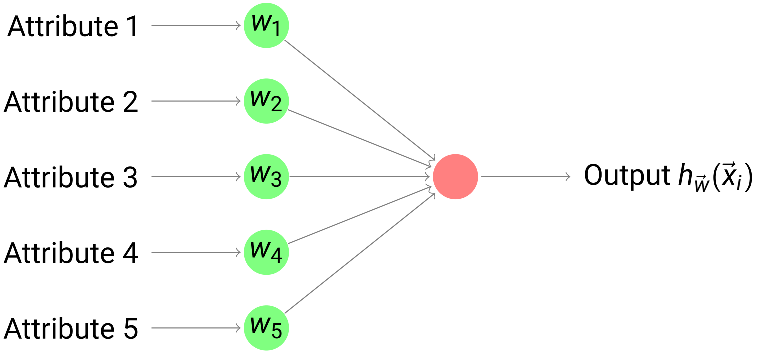
Bayes Classifiers

Multinomial Naïve Bayes

- Considers a document as a bag-of-words.
- Estimates the probability of observing word w and the prior probability $P(c)$
- Probability of class c given a test document d:

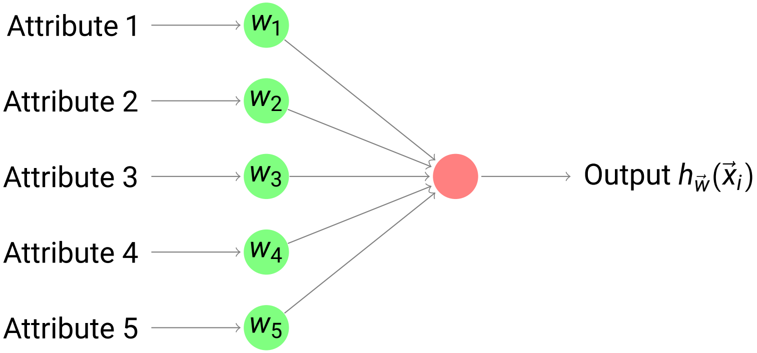
$$P(c|d) = \frac{P(c)\prod_{w\in d} P(w|c)^{n_{wd}}}{P(d)}$$

Perceptron



- Data stream: $\langle \vec{x}_i, y_i \rangle$
- Classical perceptron: $h_{\vec{w}}(\vec{x}_i) = \mathrm{sgn}(\vec{w}^T \vec{x}_i)$,
- Minimize Mean-square error: $J(\vec{w}) = \frac{1}{2} \sum (y_i - h_{\vec{w}}(\vec{x}_i))^2$

Perceptron



- We use sigmoid function $h_{\vec{w}} = \sigma(\vec{w}^T\vec{x})$ where

$$\sigma(x) = 1/(1+e^{-x})$$

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

Perceptron

- Minimize Mean-square error: $J(\vec{w}) = \frac{1}{2}\sum(y_i - h_{\vec{w}}(\vec{x}_i))^2$
- Stochastic Gradient Descent: $\vec{w} = \vec{w} - \eta \nabla J \vec{x}_i$
- Gradient of the error function:

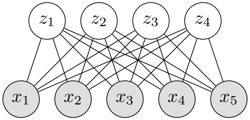
$$\nabla J = -\sum_i (y_i - h_{\vec{w}}(\vec{x}_i)) \nabla h_{\vec{w}}(\vec{x}_i)$$

$$\nabla h_{\vec{w}}(\vec{x}_i) = h_{\vec{w}}(\vec{x}_i)(1 - h_{\vec{w}}(\vec{x}_i))$$

- Weight update rule

$$\vec{w} = \vec{w} + \eta \sum_i (y_i - h_{\vec{w}}(\vec{x}_i)) h_{\vec{w}}(\vec{x}_i)(1 - h_{\vec{w}}(\vec{x}_i)) \vec{x}_i$$

Restricted Boltzmann Machines (RBMs)



- Energy-based models, where

$$P(\vec{x},\vec{z}) \propto e^{-E(\vec{x},\vec{z})}.$$

- Manipulate a weight matrix W to find low-energy states and thus generate high probability $P(\vec{x},\vec{z})$, where

$$E(\vec{x},\vec{z}) = -W.$$

- RBMs can be stacked on top of each other to form so-called Deep Belief Networks (DBNs)

Classification

Example

Data set that describes e-mail features for deciding if it is spam.

Contains "Money"	Domain type	Has attach.	Time received	spam
yes	com	yes	night	yes
yes	edu	no	night	yes
no	com	yes	night	yes
no	edu	no	day	no
no	com	no	day	no
yes	cat	no	day	yes

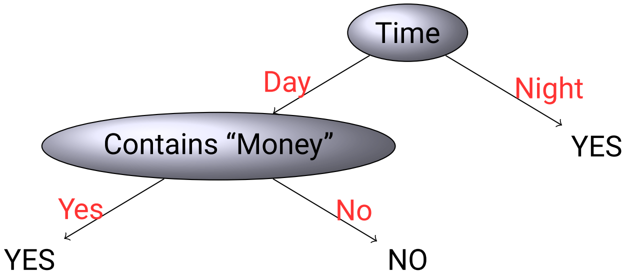
Assume we have to classify the following new instance:

Contains "Money"	Domain type	Has attach.	Time received	spam
yes	edu	yes	day	?

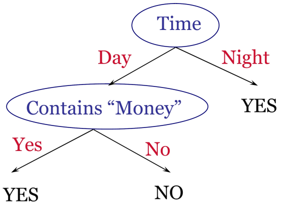
Classification

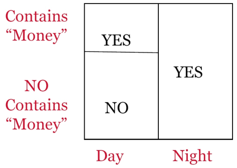
- Assume we have to classify the following new instance:

Contains "Money"	Domain type	Has attach.	Time received	spam
yes	edu	yes	day	?



Decision Trees





Basic induction strategy:

- *A* $\leftarrow$ the "best" decision attribute for next *node*
- Assign *A* as decision attribute for *node*
- For each value of *A*, create new descendant of *node*
- Sort training examples to leaf nodes
- If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Bagging

Example

Dataset of 4 Instances : A, B, C, D

Classifier 1: B, A, C, B
Classifier 2: D, B, A, D
Classifier 3: B, A, C, B
Classifier 4: B, C, B, B
Classifier 5: D, C, A, C

Bagging builds a set of *M* base models, with a bootstrap sample created by drawing random samples with replacement.

Random Forests

- Bagging
- Random Trees: trees that in each node only uses a random subset of the attributes

Random Forests is one of the most popular methods in machine learning.

Boosting

The strength of Weak Learnability, Schapire 90

A boosting algorithm transforms a weak learner into a strong one

Boosting

A formal description of Boosting (Schapire)

- given a training set $(x_1, y_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \ldots, T$
    - construct distribution D_t
    - find weak classifier

    $$h_t : X \to \{-1, +1\}$$

    with small error $\varepsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$ on D_t
- output final classifier

Boosting

AdaBoost

```
1: Initialize $D_1(i) = 1/m$ for all $i \in \{1, 2, ..., m\}$
2: **for** $t$ = 1,2,...$T$ **do**
3:     Call **WeakLearn**, providing it with distribution $D_t$
4:     Get back hypothesis $h_t : X \to Y$
5:     Calculate error of $h_t : \varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
6:     Update distribution
       $D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \varepsilon_t/(1-\varepsilon_t) & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
       where $Z_t$ is a normalization constant (chosen so $D_{t+1}$ is a
       probability distribution)
7: **return** $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} -\log \varepsilon_t/(1-\varepsilon_t)$
```

Boosting

AdaBoost

```
1: Initialize $D_1(i) = 1/m$ for all $i \in \{1, 2, ..., m\}$
2: **for** $t$ = 1,2,...$T$ **do**
3:     Call **WeakLearn**, providing it with distribution $D_t$
4:     Get back hypothesis $h_t : X \to Y$
5:     Calculate error of $h_t : \varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
6:     Update distribution
       $D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \varepsilon_t & \text{if } h_t(x_i) = y_i \\ 1 - \varepsilon_t & \text{otherwise} \end{cases}$
       where $Z_t$ is a normalization constant (chosen so $D_{t+1}$ is a
       probability distribution)
7: **return** $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} - \log \varepsilon_t / (1 - \varepsilon_t)$
```

Stacking

Use a classifier to combine predictions of base classifiers

Example

- Use a perceptron to do stacking
- Use decision trees as base classifiers

Clustering

Definition

Clustering is the distribution of a set of instances of examples into non-known groups according to some common relations or affinities.

Example

Market segmentation of customers

Example

Social network communities

Clustering

Definition

Given

- a set of instances I
- a number of clusters K
- an objective function $cost(I)$

a clustering algorithm computes an assignment of a cluster for each instance

$$f : I \rightarrow \{1, \ldots, K\}$$

that minimizes the objective function $cost(I)$

Clustering

Definition

Given

- a set of instances I
- a number of clusters K
- an objective function $cost(C, I)$

a clustering algorithm computes a set C of instances with $|C| = K$ that minimizes the objective function

$$cost(C, I) = \sum_{x \in I} d^2(x, C)$$

where

- $d(x, c)$: distance function between x and c
- $d^2(x, C) = min_{c \in C} d^2(x, c)$: distance from x to the nearest point in C

k-means

- 1. Choose k initial centers $C = \{c_1, \ldots, c_k\}$
- 2. while stopping criterion has not been met
  - For $i = 1, \ldots, N$
    - find closest center $c_k \in C$ to each instance p_i
    - assign instance p_i to cluster C_k
  - For $k = 1, \ldots, K$
    - set c_k to be the center of mass of all points in C_i

k-means++

```
1. Choose a initial center $c_1$
    For $k = 2, \ldots, K$
        select $c_k = p \in I$ with probability $d^2(p, C)/cost(C, I)$
2. while stopping criterion has not been met
        For $i = 1, \ldots, N$
            find closest center $c_k \in C$ to each instance $p_i$
            assign instance $p_i$ to cluster $C_k$
        For $k = 1, \ldots, K$
            set $c_k$ to be the center of mass of all points in $C_i$
```

Performance Measures

Internal Measures

- Sum square distance
- Dunn index $D = \frac{d_{min}}{d_{max}}$
- C-Index $C = \frac{S - S_{min}}{S_{max} - S_{min}}$

External Measures

- Rand Measure
- F Measure
- Jaccard
- Purity

Density based methods

DBSCAN

- ε-neighborhood(p): set of points that are at a distance of *p* less or equal to ε
- Core object: object whose ε-neighborhood has an overall weight at least μ
- A point *p* is *directly density-reachable* from *q* if
  - *p* is in ε-neighborhood(q)
  - *q* is a core object
- A point *p* is *density-reachable* from *q* if
  - there is a chain of points $p_1, \ldots, p_n$ such that p_{i+1} is directly density-reachable from p_i
- A point *p* is *density-connected* from *q* if
  - there is point *o* such that *p* and *q* are density-reachable from *o*

Density based methods

DBSCAN

- A *cluster* C of points satisfies
  - if $p \in C$ and q is density-reachable from p, then $q \in C$
  - all points $p, q \in C$ are density-connected
- A *cluster* is uniquely determined by any of its core points
- A *cluster* can be obtained
  - choosing an arbitrary core point as a seed
  - retrieve all points that are density-reachable from the seed

DBSCAN

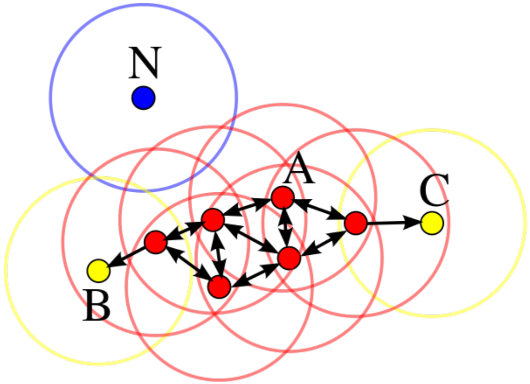


Figure: DBSCAN Point Example with μ=3

Density based methods

DBSCAN

- select an arbitrary point *p*
- retrieve all points density-reachable from *p*
- if *p* is a core point, a cluster is formed
- If p is a border point
    - no points are density-reachable from p
    - DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed