Babinkostova Notes

Michel Liao

May 2022

Contents

1 Introduction to Deep Learning and Neural Networks

- Based on this article.
- **Deep learning** is a type of machine learning (ML) that's designed to predict an output based on an input.
  - Usually used when data has a large number of inputs involved.
  - **Backpropogation** occurs when an algorithm compares its output layer to the actual output and corrects itself by going backwards through the layers and changing the parameters

2 Data Mining Essentials

- Based on this chapter.
- We extract useful patterns through **knowledge discovery databases**.

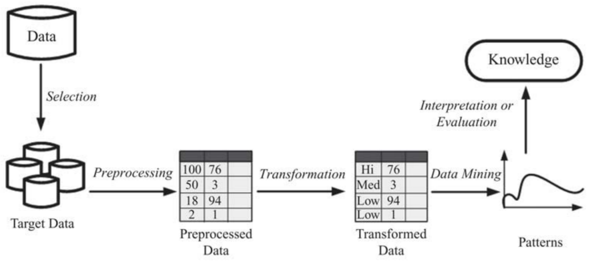

Figure 1: Knowledge Discovery in Databases (KDD) process.

2.1 Data

- We can use **vectorization** to convert text into tabular format. A well-known method is the **vector-space model.**

2.1.1 Vector Space Model

- Given a set of documents D (with a set of words in them), we can represent document i with vector d_i:

$$d_i = (w_{1,i}, w_{2,i}, \dots, w_{N,i}),$$

where $w_{j,i}$ represents the weight for word j that occurs in document i and N is the number of words used for vectorization.

  - To compute $w_{j,i}$, you can set it to 1 if it exists in a document and 0 if it doesn't.
  - You can also set it to the number of times the word j is observed in document i.
  - A more generalized approach is to use the **term frequency-inverse document frequency** (TF-IDF) weighting scheme:

$$w_{j,i} = tf_{j,i} \times idf_j,$$

where $tf_{j,i}$ is the frequency of word j in document i and idf_j is the inverse TF-IDF frequency of word j across all documents,

$$idf_j = \log_2 \frac{|D|}{|\{\text{document} \in D | j \in \text{document}\}|},$$

which is the total number of documents divided by the number of documents that contain the word j.

2.1.2 Data Quality

- We need to verify the quality of data before applying data mining algorithms to the vectorizations.
- We need to verify four data quality aspects:
    1. **Noise**, or the distortion of the data. Noise needs to be removed or alleviated before running data mining algorithms. This can be done through filtering algorithms.
    2. **Outliers**, or instances that are considerably different from other instances in the dataset. The decision to remove outliers depends on the context of the problem.
    3. **Missing Values**, or feature values that are missing. To solve this, you can remove the instances that have missing values, estimate missing values (e.g. with the most common value), or ignore missing values when running algorithms.
    4. **Duplicate data**, or multiple instances with the same feature values.

2.2 Data Preprocessing

- Data often has to be preprocessed to prepare the data for mining. The typical preprocessing steps are the following:
    1. **Aggregation.** We aggregate data when multiple features need to be combined into a single feature or the scale of the features change. E.g. instead of storing image dimensions, store the image area. This reduces data variance, and hence grants higher resistance to noise.
    2. **Discretization.** Converting continuous features to discrete ones and deciding the continuous range that is assigned to a discrete value.
    3. **Feature Selection.** Not all features gathered are useful or there is a lack of computational power. We can select a subset of features that enhance a data mining algorithm's performance.
    4. **Feature Extraction.** Converting the current set of features to a new set of features that can perform the data mining task better.
    5. **Sampling.** Often, processing whole datasets is expensive. So, we take a sample, instead:
        - **Random sampling.** In a dataset of size n, all instances have an equal probability of $\frac{1}{n}$ of being selected.
        - **Sampling with or without replacement.** We can sample with replacing a chosen instance or not.
        - **Stratified sampling.** The dataset is partitioned into multiple bins which undergo random sampling. This helps when there isn't a uniform distribution for class attribute values.

- We can sample networks by selecting a subset of their nodes and edges using the aforementioned methods or by starting with a small set of nodes (**seed nodes**) and sample the connected components they belong to; the set of nodes and edges connected to them directly; or the set of nodes and edges that are within n-hop distance from them.

2.3 Data Mining Algorithms

- Two well-established categories of data mining are **supervised learning** and **unsupervised learning**.
  - Supervised Learning. The class attribute exists and we are trying to predict the class attribute value.
  - Unsupervised Learning. The dataset has no class attribute and our task is to find similar instances in the dataset and group them to find significant patterns.

2.4 Supervised Learning

- **Training data** is in the format $(\vec{x}, y)$, where $\vec{x}$ is a vector and y is the class attribute. Supervised learning builds a model that maps $\vec{x}$ to y.
  - In other words, we want to find a mapping m such that $m(x) = y$.
  - For a test dataset, we're given $(\vec{x}, ?)$, where we can compute $m(x)$ to find our predictions.
- Supervised learning is divided into **classification**, when the class attribute is discrete, and **regression**, when the class value is continuous.

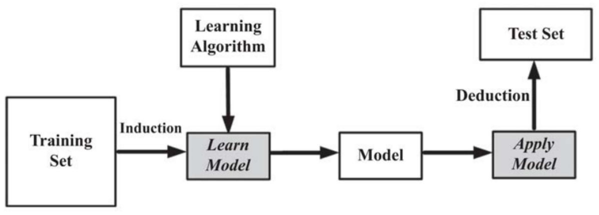

Figure 5.2: Supervised Learning.

Figure 2: Supervised Learning.

- A supervised learning algorithm is run on the training set, known as **induction**, wherein a model is generated that maps feature values to class attribute values. Then, the model is used on a test set to predict unknown class attribute values (**deduction**).

2.4.1 Decision Tree Learning

- Given data about characteristics of Twitter individuals, we can predict whether or not they're influential with decision trees.

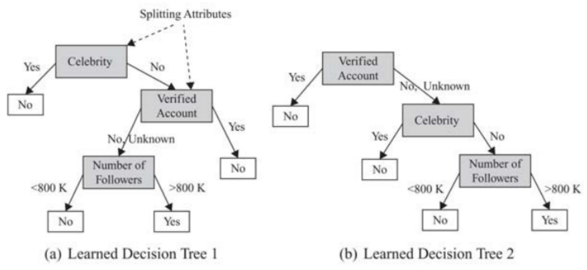

Figure 3: Decision Trees Learned from Data Provided.

  - Each nonleaf node represents a feature and each branch represents a value that feature can take. An instance is classified by following a path that starts at the root node and ends at a leaf.

- Different heuristics can generate different decision trees that all correctly predict the class attribute values.

- When selecting features, we prefer features that partition the set of instances into subsets that are **pure**. A pure subset has instances that all have the same class attribute value. We can measure purity with **entropy**:

  - Over a subset of training instances T with binary class attribute (values $\in \{+, -\}$), the entropy of T is defined by

$$\text{entropy}(T) = -p_+ \log p_+ - p_- \log p_-,$$

    where p_+ and p_- are the proportions of instances with a + and − class attribute value in T, respectively.

  - In a pure subset, all instances have the same class attribute value and the entropy is 0.

2.4.2 Naive Bayes Classifier (NBC)

- Given two random variables X and Y, **Bayes theorem** states that

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}.$$

  - In NBC, Y represents the class variable and X represents the instance features.

- Let $X = (x_1, x_2, x_3, \ldots, x_m)$, where x_i represents the value of feature i. Let $Y = \{y_1, y_2, \ldots, y_n\}$ represent the values Y can take. We can find the class attribute value of instance X with

$$\arg\max_{y_i} P(y_i|X)^{[1]}. \tag{1}$$

- Based on Bayes theorem, we have that

$$\arg\max_{y_i} P(y_i|X) = \frac{P(X|y_i)P(y_i)}{P(X)}.$$

  Note that $P(X)$ stays constant, so we can ignore it when trying to maximize (1).

- We assume conditional independence, i.e. given the class attribute, the other features are conditionally independent, to find

$$P(X|y_i) = \prod_{j=1}^{m} P(x_j|y_i).$$

- Thus,

$$P(y_i|X) = \frac{(\prod_{j=1}^{m} P(x_j|y_i))P(y_i)}{P(X)}.$$

2.4.3 Nearest Neighbor Classifier

- k-nearest neighbor, or kNN, uses the k nearest instances, called neighbors, to perform classification.

- The instance being classified is assigned the class attribute value that the majority of its k neighbors are assigned.

- The algorithm, requiring instance i, a dataset of real-value attributes, k number of neighbors, and a distance measure d, is as follows:

  1. Return class label for instance i
  2. Compute k nearest neighbors of instance i based on distance measure d
  3. l = the majority class label among neighbors of instance i. If there is more than one majority label, select one randomly.
  4. Classify instance i as class l.

- The choice of k greatly influences the label of an instance being predicted.

[1] The function arg max finds the argument that will produce a maximum for another function. Say we want to find $\arg\max(g(x))$, where $g(x) = x^2$ for $1 \leq x \leq 5$. Then, $\arg\max(g(x)) = 5$. Read more about arg max here.

2.4.4 Classification with Network Information

- To find out more, go to this PDF's corresponding section.

2.4.5 Regression

- In classification, class attribute values are discrete. In regression, class attribute values are real numbers.
- Input to the regression method is a dataset where the attributes are represented using $X = x_1, x_2, \ldots, x_m$ (**regressors**) and class attribute is represented using Y.

Linear Regression

- We assume that the class attribute Y has a linear relation with the regressors (features) X by considering the linear error ϵ.

$$Y = XW + \epsilon,$$

  where W represents the vector of regression coefficients.

- We estimate W by using the training dataset and its labels Y such that ϵ is minimized.
- Commonly, we use least squares to minimize ϵ:

$$\epsilon^2 = ||\epsilon^2|| = ||Y - XW||^2.$$

- To minimize ϵ, we compute the gradient and set it to zero to find the optimal W:

$$\frac{\partial ||Y - XW||^2}{\partial W} = 0.$$

- Then,

$$W = (X^T X)^{-1} X^T Y.$$

Logistic Regression

- To find out more, go to this PDF's corresponding section.

2.4.6 Supervised Learning Evaluation

- Supervised learning algorithms are often evaluated through a **training-testing** framework where a training dataset trains the model and the model is then evaluated on a test dataset.
- We usually divide the training set into train/test sets. We can divide the training set using the following methods:

  - **Leave-one-out training**: Partition the training set into k equally sized **folds** and use all folds but one to train.
  - **k-fold cross validation**: Divide the training set into k equally sized sets and run the algorithm k times. In round i, use all the folds but fold i for training and fold i for testing. The average performance of the algorithm over k rounds measures the **generalization accuracy** of the algorithm.

- For classification, the class attribute values are discrete, so we can use accuracy to evaluate the classifier:

$$\text{accuracy} = \frac{c}{n},$$

  where c is the number of instances which the labels were correctly predicted and n is the size of the test dataset.

- For regression, we check if the predictions are highly correlated with the **ground truth** using correlation analysis or fitting lines to ground truth and prediction results and checking if the lines are close.

2.5 Unsupervised Learning

- Unsupervised learning is the unsupervised division of instances into groups.
- In clustering, data is often unlabeled. Instances are put into clusters based on their distance to other instances.
- For continuous features, the most popular distance measure is **Euclidean distance**:

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2},$$

  where $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ are n-dimensional feature vectors in $\mathbb{R}^n$.

2.5.1 Clustering Algorithms

Partitional Algorithms

- Partitional clustering algorithms partition the dataset into a set of clusters. Each instance is assigned to a cluster exactly once, and no instance remains unassigned.
- k-means is a well-known example of a partitional algorithm.
  - The algorithm starts with k initial centroids (randomly chosen instances from the dataset).

- Then, we assign each instance to one of the clusters based on its distance to the centroid of each cluster.
- After all instances are assigned, the centroids are recomputed by taking the average of all instances inside the clusters. This is repeated until convergence.
- k-means implementations try to minimize an **objective function.** For example, the squared distance error:

$$\sum_{i=1}^{k} \sum_{j=1}^{n(i)} ||x_j^i - c_i||^2,$$

  where x_j^i is the jth instance of cluster i, $n(i)$ is the number of instances in cluster i, and c_i is the centroid of cluster i.
  * This process stops once the difference between consecutive objective function values is bounded by some small ϵ.
- k-means is highly sensitive to the initial k centroids and different clustering results can be obtained depending on the initial centroids.
  * We can mitigate this problem by running k-means multiple times and selecting the clustering assignment that is observed most often or is most desirable based on an objective function.

2.5.2 Unsupervised Learning Evaluation

- It's hard to evaluate clustering because ground truth isn't available. But, we can measure the **cohesiveness** or **separateness** of clusters.

Cohesiveness

- In cohesive clusters, instances inside the clusters are close to each other:

$$\text{cohesiveness} = \sum_{i=1}^{k} \sum_{j=1}^{n(i)} ||x_j^i - c_i||^2,$$

  which is the square distance error (SSE).

Separateness

- We want clusters that are well-separated from one another. In other words, we want cluster centroids far from the mean of the entire dataset:

$$\text{separateness} = \sum_{i=1}^{k} ||c - c_i||^2,$$

  where $c = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the centroid of all instances and c_i is the centroid of cluster i.

Silhouette Index

- The silhouette index combines both cohesiveness and separateness.
- Let $a(x)$ denote the average distance between instance x of cluster C and all other members of C:

$$a(x) = \frac{1}{|C| - 1} \sum_{y \in C, y \neq x} ||x - y||^2.$$

- Let $G \neq C$ denote the cluster that is closest to x in terms of the average distance between x and members of G. Let $b(x)$ denote the average distance between instance x and instances in cluster G:

$$b(x) = \min_{G \neq C} \frac{1}{|G|} \sum_{y \in G} ||x - y||^2.$$

- We want distance between instances in the same cluster ($a(x)$) to be smaller than the distance between different clusters ($b(x)$), or $a(x) < b(x)$.

$$s(x) = \frac{b(x) - a(x)}{\max(b(x), a(x))}$$
$$\text{silhouette} = \frac{1}{n} \sum_{x} s(x).$$

- The silhouette index takes values between $[-1, 1]$. The best clustering happens when $\forall x,\ a(x) \ll b(x)$ (silhouette ≈ 1).

3 Other ML Resources

- An Introduction to Machine Learning.
- What is a Convolutional Neural Network?