Università degli Studi di Padova

Department of Information Engineering

Master Thesis in Computer Engineering

Sentiment Analysis on Online Automotive Forums

Supervisor
Nicola Ferro

Master Candidate
Giuseppe Ravagnani

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetuer. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

Quisque facilisis erat a dui. Nam malesuada ornare dolor. Cras gravida, diam sit amet rhoncus ornare, erat elit consectetuer erat, id egestas pede nibh eget odio. Proin tincidunt, velit vel porta elementum, magna diam molestie sapien, non aliquet massa pede eu diam. Aliquam iaculis. Fusce et ipsum et nulla tristique facilisis. Donec eget sem sit amet ligula viverra gravida. Etiam vehicula urna vel turpis. Suspendisse sagittis ante a urna. Morbi a est quis orci consequat rutrum. Nullam egestas feugiat felis. Integer adipiscing semper ligula. Nunc molestie, nisl sit amet cursus convallis, sapien lectus pretium metus, vitae pretium enim wisi id lectus. Donec vestibulum. Etiam vel nibh. Nulla facilisi. Mauris pharetra. Donec augue. Fusce ultrices, neque id dignissim ultrices, tellus mauris dictum elit, vel lacinia enim metus eu nunc.

Contents

1
Introduction

2
State of the Art

2.1 Sentiment Analysis

From the introduction of online social media, people generate continuously vast quantities of contents such as texts, reviews, comments, videos, pictures. This enormous quantity of data offers the possibility to access and understand users' perspective about topics of interest, and this can be exploited, for instance, to make market predictions, business choices, predict outcomes of political elections. *Sentiment Analysis* is an active area of research motivated to improve the automated recognition of sentiment expressed in text [1].
From the whole scenic of online social media, Twitter * is the most used to achieve contents, due to a very active community and the simplicity to get data from Twitter's APIs (Application Programming Interface).

2.1.1 Twitter Sentiment Analysis

From recent statistics dated 2018, Twitter counts 326 million monthly active users, 100 million daily active users, which send 500 million tweets per day (80% from mobile). That large amount of users and messages motivates the fact that most of the researches are made using Twitter's data, so usually this area is referenced as *Twitter Sentiment Analysis* (TSA).

*Twitter is an online micro-blogging social network, where people interact with short messages, known as ”tweets”. It was created in March 2006 by Jack Dorsey, Noah Glass, Biz Stone, and Evan Williams and launched in July of that year.

In addition to the plenty of data, Twitter becomes a case study because tweets often express users' opinions about a topic of interest. Twitter Sentiment Analysis has been applied to mine information from users, in order to explain and make predictions on different social phenomena, for instance in [2] researches studied the outcomes of political elections, in [3] stock market movements and in [4] product sales. Sentiment Analysis has also been applied without involving Twitter's data, for instance in [5] the goal was to monitor the software deployment.

We noticed that Sentiment Analysis has a wide variety of applications, however even if the attention of researches and the growing of the literature, the performances of state-of-the-art approaches are very poor, due to the difficulties on interpreting the natural language.

Sentiment analysis can be summarized as the automatic classification task of sentiment polarity expressed in text (for instance with classes positive, negative and neutral), identifying the stance about a topic, opinion holder identification, and sometimes considering various aspects of a topic. The sentiment polarity classification problem can be modeled in different ways, depending on the task. It can be considered a binary classification (positive/negative) or three-way classification (positive/negative/neutral), but sometimes a more fine-grained classification is needed, considering also slightly positive and slightly negative classes. Sentiment polarity classification may be also domain specific: for instance texts classified as positive express in favor of a certain brand, or a candidate of the elections, or optimism about social issues.

Approaches for sentiment analysis can be categorized in two main classes [1]:

- The first class contains methods that involves the use of a lexicon of opinion-related terms, with a scoring method to evaluate the sentiment. These methods are used in unsupervised applications, but they achieve low performances, since they are unable to consider some important part of the sentence, for instance context, novel vocabulary, indicators for sentiment expressions;
- The second class is represented by methods based on feature representation of texts, that involves the use of machine learning approaches to derive relationships between feature values and sentiment using supervised learning. These methods require a large set of labeled training samples to train a model, but they have the limit that if they are thought for domain specific, they does not work on broader applications. In the next sections I will focus on these methods.

In general, the effectiveness of the information derived from TSA campaigns, are critically dependent on the approach to evaluate the sentiment expressed by users. Some approaches are directly taken from literature and they are thought for more established social media, like product reviews and web forums. However, several characteristics make hard to apply the same techniques on tweets. These characteristics include brevity of texts (less than 140 characters length), adoption of novel language, twitter specific communication elements, strong sentiment class imbalance (usually most of tweets are labeled as neutral), and stream based tweets generation. Considering these characteristics is crucial to achieve adequate performances from TSA models. For these reasons, some researches focus more on sentiment analysis on other social media, for instance on Facebook* status updates that can contain up to 5000 characters. Other social media have the characteristic of long texts, but strong imbalance among classes: for example online reviews are predominantly positive or negative, while web forums are mostly neutral. In this review I focus more on Twitter Sentiment Analysis, but the same techniques can be applied (eventually with some differences) on other types of social media. In the next paragraph I'm going to discuss some techniques to address issues on tweets due to their brevity.

2.1.2 Techniques to address tweets' issues

As previously said, tweets are text messages limited to 140 characters, and their brevity impact the performance of sentiment analysis techniques, since there are just few terms that capture information about user's sentiment. This fact implies also the very sparsity of feature vectors that represent tweets. The length limitation, motivates users to adopt novel form of expression, slang, acronyms and emoticons. Moreover, novel terms are continuously evolving, so some of them may not be considered in all TSA approaches.

One class of techniques addresses this issue propagating known sentiment information, coming from vocabulary of emoticons and lexicon of sentiment terms, through tweets to identify new expressions related to sentiment, looking for instance the co-occurrence or proximity of tuples of terms. Once propagated, the vocabulary is enlarged, so a bigger set of lexicon sentiment information may improve effectiveness of TSA approaches [6].

Another approach is based on feature expansion, adding more related content or forming combination of the current one. In this task, some lexical resources like WordNet are useful to include words' synonyms, or add semantic concepts related to named entities [7].

Twitter has communication elements that are not present in other social media, and can contain information about sentiment polarity. These elements also increase the vocabulary and

*Facebook is a social network founded by Mark Zuckerberg aling with fellow Harvard College students and roommates Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. Facebook was launched on February 4 2004.

so they can complicate sentiment analysis. Some of these components are user mentions (@username) used for direct the tweet to the mentioned user, hashtags (#hashtag) used to connect the tweet to other that threat the same arguments. Especially the second ones sometimes contains information about sentiment, for instance hashtags like #sad or #bad may express negative feelings, or contrary #good or #wonderful may express positive ones. For these facts it is important to include these elements in the sentiment decision, paying attention that not always they are not entirely included in the body of the tweet.
One class of techniques involves the just discussed Twitter-specific communication elements. It consists on removing, grouping or correcting them in a preprocessing task. This lead to complexity reducing and consequently reducing the feature sparsity. Some approaches removes any occurrences of hashtags, while some others replace them with common tokens, for instance *"USER_MENTION"*. In some other cases, syntactic exaggerations (like multiple exclamation marks) are simplified and misspellings are corrected. Also emoticons are handled for instance grouping positive emoticons (like *:)* or *:-)*) and the same for negative ones. Some other approaches include these elements in the feature vectors.

In the next paragraph I will discuss how sentiment class imbalance is handles.

2.1.3 Techniques to address sentiment class imbalance

As previously said, tweets are predominantly neutral (while for instance product reviews are generally either positive or negative), so usually TSA faces class imbalance issues that can affect classification tasks. Large class imbalances are problematic especially for machine learning because they induce bias, so learned models have preferences on predicting neutral class (in the case of TSA).
One strategy to overcome these issues consists on expanding the training set utilized to train the machine-learning classifiers. In this way the set of comments will contain various form of expressions of sentiment expressed in tweets, so it may improve the classification performance.
Another approach consists on applying multiple classifiers in an ensemble. Since classifiers' performance may vary, the ensemble model the overall classification more stable. A similar approach consists on applying firstly a subjectivity classifier (responsible of select the subject), then a sentiment classifier (responsible of the sentiment classification).

In the next paragraph I will discuss how to handle temporal dependencies.

2.1.4 Techniques to address temporal dependencies

One issue on performing TSA on streaming is that topic and language may change rapidly, and model may not be calibrated on needed ones. Tweets sometimes are also temporary dependent, since a user can express his opinion using more tweets. This characteristic of communication can be shared with other types of social media, for instance web forums. In this class, the temporal dependency can be shown on quotes: usually happens that a user expressed his opinion on something someone on the conversation said. In these cases the sentiment of the comment is dependent by the referenced text, to it may be useful to take in account both the comment and the reference to decide the sentiment. The majority of sentiment classifiers threat tweets as independent instances, sometimes losing information about ordering or temporal references.

2.1.5 Common approaches

Sentiment analysis is a composition of diverse problems, in fact some steps are needed to perform opinion mining on given texts, since texts are coming from different resources and different formats. Data acquisition and preprocessing are mandatory sub-tasks required for opinion mining.

Data acquisition is an essential task for collecting the dataset, since online resources generally don't satisfy all requirements, for instance the focused domain, or the social media where data are extracted. For this task, big social media such as Twitter and Facebook share APIs for collecting public data, contrary for other online resources such as forums, *web scraping* [†] is a useful, but less comfortable technique.

Raw data collected with in the previous task must be preprocessed, since they are full of noise (useless parts of sentences that do not give any information). Some common steps are: tokenization, stop word removal, stemming, parts of speech (POS) tagging, and feature extraction and representation [8]. Tokenization is used to break a sentence into words, phrases, symbols, or other meaningful tokens by removing punctuation marks. Stopwords are also dropped since they don't carry information so they don't contribute on analysis. Stemming is the process to transform a word into its root form by dropping the suffix. POS tagging is

[†] Web scraping is a technique for extracting data from websites. Web scraping software access the websites through Hypertext Transfer Protocol, sometimes using automatic crawlers, then the downloaded page is parsed in order to extract the needed information.

performed to recognize different parts of speech in the text. Due to the characteristic sparsity and noise of the data in natural language processing problems, feature selection is an important preprocessing step, critical for the final performance.

Sentiment classification, as said before, can be performed using machine learning approaches as well as lexicon based approaches. Machine learning approaches can be divided in supervised learning and unsupervised learning methods: for what concerns supervised ones, common algorithms are Decision Trees, Support Vector Machines (SVM), Neural Networks (NN), Naïve Bayes. Lexicon based approaches uses either dictionaries or corpus based approaches. Dictionary methods use existing resources collecting information about words along with their positive or negative sentiment strength. Corpus based approaches, instead, are based upon the probability of occurrence of a sentiment word along with positive or negative set of words by performing search on texts taken from the web. The two different approaches are summarized in Figure 2.1.

Since this work is focused on machine learning approaches, the main steps, common to most text classification algorithms, deserve a better explanation.

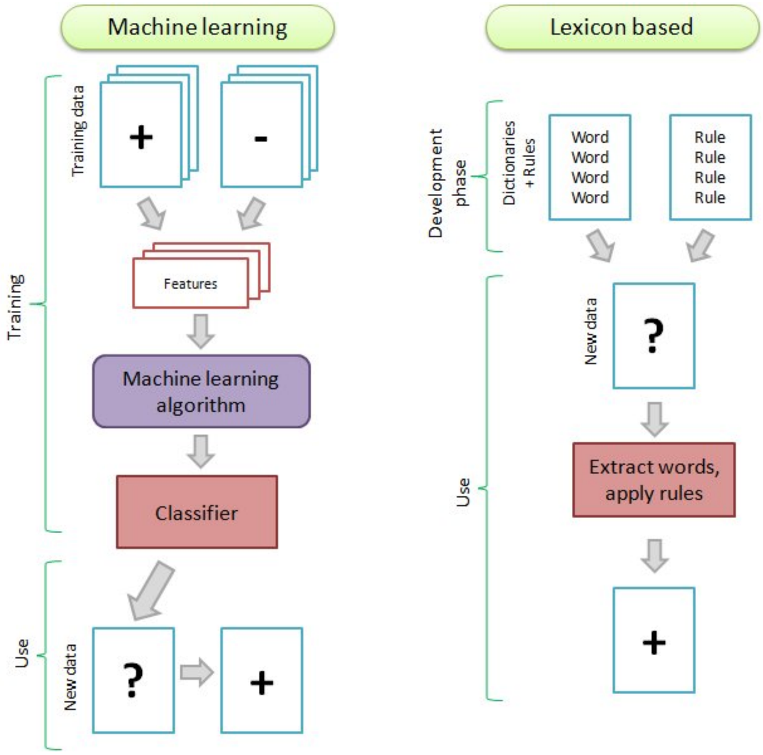

Figure 2.1: Comparison between machine learning and lexicon-based approaches on sentiment analysis

Tokenization

Given a characters sequence, tokenization is the task of splitting it into pieces, called *tokens*, while also dropping some useless characters like punctuation. Below an example:

Input: Friends, Romans, Countrymen, lend me your ears;

Output: ['Friends', 'Romans', 'Countrymen', 'lend', 'me', 'your', 'ears']

It is important to give some definitions [9]: a token consist in a sequence of characters grouped together in a useful semantic unit, a *type* is the class of all tokens containing the same character sequence, a *term* is a type that is included in a reference dictionary. The major question on tokenization is to select which are the correct tokens to use, for instance, for "*aren't*", which is the more correct tokenization between "*arent*", "*are n't*" or "*aren t*? A simple strategy is to split on all non-alphanumerical characters, but in this case "*aren t*" does not look correct In general tokenization is language-specific, so every language has its own rules to make a useful tokenization.

Stopwords removal

A stopword is a commonly used word, such as "the", "a", "an" or "in" that do not contribute on giving information to a text, so usually are removed. It is possible to achieve language-specific resources to get the list of stopwords for a language. An example of this task is below:

Input: "This is a sample sentence, showing off the stop words filtration."

Tokenized: ['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']

Stopwords removal: ['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']

Stemming

For grammatical reasons, texts use different forms of a word, such as "organize", "organise", "organizing", so it will be useful to refer all these versions to a common one. The goal of stemming is to reduce the set of possible words by transforming all words to their common base form, for instance:

am, are, is ⇒ be

car, cars, car's, cars' ⇒ car

The result applied to a complete sentence will be:

Input: ”the boy's cars are different colors”

Output: ”the boy car be differ color”

Stemming is usually an heuristic process that respects some linguistic rules that cut off suffixes of the words, and sometimes the affixes. A similar but more complex method is called *lemmatization*. It consists on the same task of stemming, but instead of using heuristics rules, it uses dictionaries and morphological analysis of the words, so it aims to remove inflectional endings only and to return the base or dictionary form, also called *lemma*. An example that shows the difference between stemming and lemmatization can be simply shown on what happens to the word ”saw”: the stemming returns just ”s”, whereas lemmatization returns ”see”.

Parts Of Speech tagging

Part of speech tagging (POS tagging) is the process of marking up a word in a text as corresponding to a particular part of speech, based on both its definition and its context. Identifying parts of speech is more complicated than simply mapping words to their part of speech tag, because usually words may have different POS tags with respect to the context, for instance:

She saw a *bear*.

Your efforts will *bear* fruit.

The same word ”bear” has two complete different senses with respect to the context, moreover, it has two different parts of speech: in the first sentence it is a noun, while in the second one it is a verb.

To define POS tagging there are essentially two types of taggers. Rule-based taggers use contextual information to assign tags to unknown or ambiguous words. Disambiguation is done by analyzing the linguistic features of the word, its preceding and following ones and other aspects. For instance if the preceding word is an article, the current word must be a noun. Stochastic taggers consist on models that incorporates frequency os probability. In [10] is proposed a comparison between rule-based taggers and stochastic taggers.

Features extraction and representation

Especially for machine learning text classification techniques, it is fundamental to translate a document into a numerical vector, in a process called *vectorization*, to be processed by machine learning algorithms. Firstly it is mandatory to define the feature to be vectorized, that represent the document in a multidimensional space. The most implemented feature extraction technique is the *N-Gram*. It consists on extracting a set of n-words which occurs ”in that order” in the document. Commonly in a feature representation are used 1-gram (which is just the list of words), 2-gram, 3-gram also together, in order to extract more information in the text.
An example of 2-gram:

Input: ”After sleeping for four hours, he decided to sleep for another four.”

Output: [’After sleeping’, ’sleeping for’, ’for four’, ’four hours’, ’four he’ ’he decided’, ’decided to’, ’to sleep’, ’sleep for’, ’for another’, ’another four’]

An example of 3-gram:

Input: ”After sleeping for four hours, he decided to sleep for another four.”

Output: [’After sleeping for’, ’sleeping for four’, ’four hours he’, ’hours he decided’, ’he decided to’, ’to sleep for’, ’sleep for another’, ’for another four’]

Once defined the set of features it must be represented in a multidimensional space, where each feature is translated in a number. For this task there are several techniques [11]:

Bag of Words (BoW) is a simplified version of Term Frequency. It is used in several domains such as computer vision, natural language processing, spam filters as well as text classification. In BoW, a body of text is thought of like a bag of words. These words in a matrix are not sentences which structure sentences and grammar, and the semantic relationships between words and their order is ignored. In this model just the presence of a word matters. The list of used words is processed looking at the entire text in a preliminary phase. Below an example:

Input: “As the home to UVA’s recognized undergraduate and graduate degree programs in systems engineering. In the UVA Department of Systems and Information Engineering, our students are exposed to a wide range of range”

Bag of Words: ['As', 'the', 'home', 'to', 'UVA's', 'recognized', 'undergraduate', 'and', 'graduate', 'degree', 'program', 'in', 'systems', 'engineering', 'in', 'Department', 'Information', 'students', 'are', 'exposed', 'wide', 'range']

Bag of Features: [1,1,1,3,2,1,2,1,2,3,1,1,1,2,1,1,1,1,1,1]

In Bag of Words with a vocabulary of size $|\Sigma|$, every word is encoded in a vector of size $|\Sigma|$, with 1 at index corresponding to the word, and 0 in every other indexes. This model is the cause of issues due to words position in the sentence: the phrases "this is good" and "is this good" have the same BoW representation. Term Frequency is similar to Bag of Words, that consists on counting the number of occurrences of each word instead of just the presence.

Term Frequency-Inverse Document Frequency (TF-IDF) is a method used in conjunction with Term Frequency in order to decrease the effect of commonest words (that usually do not carry information). IDF assigns a higher weight to words with either high or low frequencies term in the document. TF-IDF is the combination of both Term Frequency and Inverse Document Frequency, and it is calculated as

$$TFIDF(w,d) = TF(w,d) * log(\frac{N}{df(t)})$$

Where N is the number of documents and $df(t)$ is the number of documents that contain the term t. Although TF-IDF overcomes to the issue of commonest words, it keeps suffering the fact that similar words are accounted for similarity since all words are independent one another.

Other techniques, called Word Embedding, such as Word2Vec [12] aim to vectorize words instead of the entire document, but these methods are not used in this work, so I'm not going to describe them.

2.1.6 Applications

Sentiment analysis can be used for a wide range of applications [13]. The most general application is in e-commerce activities, since websites allow users to submit their experience about shopping and product qualities. They provide also summary of the product by assigning rates or scores that customers can easily see. Sentiment analysis helps these websites

by converting dissatisfied customers into promoters by analyzing the huge amount of opinions.
Voice of Market (VOM) is about determining what customers are feeling about products or services of competitors. Accurate sentiment classification from voice of market involving also temporal information helps in gaining competitive advantage and new product development. Detection of real-time sentiment information may help on planning marketing campaigns and strategies, improving product features in order to satisfy as much customers as possible.
Voice of the Customer (VOC) consists on knowing what a customer is saying about a product or service, analyzing his reviews and feedback. VOC is a key element of Customer Experience Management. Extracting such information may help on understand what are the features that customers want on a specific product and what are the ones that do not want, for instance costs.
Brand Reputation Management is about managing the reputation in the market. Customers' opinion may enhance or damage brand reputation, so it is important to keep it monitored. Brand information, nowadays, are not just in advertising, public relations, but also in many conversation in online forums or in social media. Sentiment analysis can help brands on knowing how their products, services or the brands themselves are perceived by people and customers.

2.2 Sentiment Analysis Datasets

In this section I'm going to describe some sentiment analysis dataset of different domains. That can be useful in order to compare the proposed ones with the final dataset utilized in this work.

Pharma [14]

This dataset provides patients reviews about drugs. Reviews are grouped into the three aspects: benefits, side effects and overall comment. Additionally, ratings are available concerning overall satisfaction as well as a five step side effect rating and five steps effectiveness rating. The data was obtained by crawling online pharmaceutical review sites.
The attributes of the dataset are the followings:

- urlDrugName (categorical): name of drug

- condition (categorical): name of condition
- benefitsReview (text): patient on benefits
- sideEffectsReview (text): patient on side effects
- commentsReview (text): overall patient comment
- rating (numerical): 10 star patient rating
- sideEffects (categorical): 5 step side effect rating
- effectiveness (categorical): 5 step effectiveness rating

Some examples of comments of different sentiments:

- positive: "*I was used to having cramps so badly that they would leave me balled up in bed for at least 2 days. The Ponstel doesn't take the pain away completely, but takes the edge off so much that normal activities were possible. Definitely a miracle medication!!*"
- neutral: "*I'm taking Gabapentin to treat nerve disorder. My nerves are inflamed and causes to experience pelvic pain and other pelvic disorders. The Gabapentin sometimes works but I do experience relapse. For the migraines I find that the 800mg of Ibprofen works better with no side effect for my migraines, but the maxalt has a better track record.*"
- negative: "*after taking propecia for over a year, starting when i was 20 years of age my hair continued to thin, and i noticed no significant benefits*"

US Airline [15]

Sentiment analysis dataset about problems of major U.S. airline. Tweets acquired from February 2015.

Some meaningful attributes of the dataset are the followings:

- airline_sentiment: sentiment
- airline_sentiment: confidence
- airline: subject airline company
- name: user's username

- text: text of the comment
- tweet_created: timestamp of the tweet

Some examples of comments of different sentiments:

- positive: ”*@VirginAmerica plus you've added commercials to the experience... tacky.*”
- neutral: ”*@VirginAmerica What @dhepburn said.*”
- negative: ”*@VirginAmerica it's really aggressive to blast obnoxious ”entertainment” in your guests' faces &amp; they have little recourse*”

IMDB [16]

This dataset is a collection of movie reviews, along with their associated binary sentiment polarity labels. It contains balanced positive and negative instances (no neutral class in this dataset). In the collection, no more than 30 reviews are present for any given movie, in order to avoid correlation between reviews of the same movie.

The attributes of the dataset are the followings:

- text: text of the review
- sentiment: the sentiment of the review

Some examples of comments:

- positive: ”*If you like adult comedy cartoons, like South Park, then this is nearly a similar format about the small adventures of three teenage girls at Bromwell High. Keisha, Natella and Latrina have given exploding sweets and behaved like bitches, I think Keisha is a good leader. There are also small stories going on with the teachers of the school. There's the idiotic principal, Mr. Bip, the nervous Maths teacher and many others. The cast is also fantastic, Lenny Henry's Gina Yashere, EastEnders Chrissie Watts, Tracy-Ann Oberman, Smack The Pony's Doon Mackichan, Dead Ringers' Mark Perry and Blunder's Nina Conti. I didn't know this came from Canada, but it is very good. Very good!*”

- negative: "*Story of a man who has unnatural feelings for a pig. Starts out with a opening scene that is a terrific example of absurd comedy. A formal orchestra audience is turned into an insane, violent mob by the crazy chantings of it's singers. Unfortunately it stays absurd the WHOLE time with no general narrative eventually making it just too off putting. Even those from the era should be turned off. The cryptic dialogue would make Shakespeare seem easy to a third grader. On a technical level it's better than you might think with some good cinematography by future great Vilmos Zsigmond. Future stars Sally Kirkland and Frederic Forrest can be seen briefly.*"

In table 2.1 are shown some statistics of proposed datasets.

Dataset	Num. instances	Positive	Neutral	Negative
Pharma	5009	15.6%	73.3%	11.1%
U.S. Airline	14641	16.1%	21.2%	62.7%
IMDB	50.000	50%	0%	50%

Table 2.1: Dataset statistics

The proposed datasets present some differences. Since IMDB dataset is thought for sentiment analysis competitions, it is built in order to simplify the classification, setting the data balanced (50% positives - 50% negatives). Another characteristic of IMDB is that it does not contain the neutral class: as explained before, product reviews social media generally contain just positives and negatives comments, so in this case the neutral class has been omitted. Apart from IMDB, data imbalance issues affect both of the other datasets: Pharma contains lot of neutral comments (73.3%), while U.S. Airline contains mostly negative ones (62.7%). Data imbalance affects real world data, so it is an issue that must be taken in consideration. An important difference between all of the three datasets consists on the language. While tweets of U.S. Airline are very concise (due to the characters limit), movie reviews are long texts. Long texts may contain sentences that supports different sentiments, for instance an user may express positive sentiment to one aspect of a movie (in case of IMDB), while criticize some other aspects. These cases are more difficult to be handled, since datasets should contain one field for each aspect to be considered, and the corresponding sentiment. In general, the presented cases show a very simple and informal language for what concerns tweets of U.S. Airline, while a long, well expressed and full of information texts for Pharma and IMDB. Lastly, Pharma dataset contains also medical-specific terms.

2.3 Supervised Learning

Since the goal of this work is to apply some machine learning techniques for sentiment analysis, it is needed an introduction to machine learning in a technical way and the most commonly used algorithm utilized in supervised learning, and also in this review.

In general, machine learning can be considered a subfield of artificial intelligence, since algorithms in some sense ”learn” from data. The core function of machine learning is to automatically find a good predictor based on past experiences. While ”learning by memorization” approach is sometimes useful, it lacks an important aspect of learning systems, which is the generalization. To achieve generalization, the learner should scan input data and learn some ”rules”, so when new data are provided, it can output some meaningful result. While human learners can rely on experience to filter out some meaningless inputs, on machine learning data must be well defined, and based on principles that protect the learner from reaching senseless conclusions.

Before going in detail, it is important to understand why we need machine learning. There are tasks that humans and animals do everyday, where it is too difficult to extract a well defined program, for instance driving, speech recognition, image understanding, or tasks that humans cannot do, for instance analysis and understanding of large and complex data sets. Other tasks need adaptivity on input data, feature that programmed tools cannot achieve, for instance handwritten text decoding, or the just described sentiment analysis.

Machine learning is, in general, a wide domain, so it is branched in several subdomains with respect to the different approaches and goals. Learning can be divided in supervised and unsupervised, depending on the fact that ground truth examples are provided or not. In unsupervised learning, clustering and pattern recognition are the most famous techniques. Supervised learning, instead, can be viewed as learning from examples, where training samples contain information that test ones do not. From here on I will focus on just supervised learning.

2.3.1 Supervised Learning Model Definition

The fist step is to define a statistical learning model [17].

- Learner's input:
  - Domain set: an arbitrary set X of samples that we wish to label. Usually samples, also called *instances*, are represented by vectors of *features*.

  - Label set: the set Y of all possible labels, for instance $Y = \{0, 1\}$
  - Training data: the set $S = ((x_1, y_1), \dots, (x_n, y_n))$ finite sequence of labeled points on $X \times Y$. That is the input provided to the learner.

- Learner's output: the learner is requested to output a *prediction rule* $h : X \to Y$, called also *hypothesis* or *classifier*. It can be used to predict labels of new points.

- Data generation model: we assume that instances are generated by some probability distribution D over X, that represents the environment. It is important that the learner should not know anything about the distribution. We assume also that there is a correct labeling function $f : X \to Y$, and that $f(x_i) = y_i, \forall i$, almost initially. The labeling function is unknown, since it is what the learner aims to find. Summarizing, each pais of S is sampled according to D and then labeled by f.

- Measure of success: we define the *error of the classifier* the probability that it does not predict the correct label of a point generated according to the generative distribution. Formally, it is defined as:

  $$L_{D,f}(h) \overset{\text{def}}{=} P_{x \sim D}(h(x) \neq f(x))$$

  That is, given a predictor h, the probability that randomly chosen an observation x according to D, we have that $h(x) \neq f(x)$.
  $L_{(D,f)}(h)$ is called *generalization error*, or *true error* of h.

2.3.2 Empirical Risk Minimization

The learner that receives in input a training set S, outputs an hypothesis $h_s : x \to Y$. The algorithm searches the hypothesis h_s that minimizes the error, but since both D and f are not known, it minimizes the so called *training error* L_s defined as:

$$L_s(h) \overset{\text{def}}{=} \frac{|\{i \in [m] : h(h_i) \neq y_i\}|}{m}$$

for $[m] = 1, \dots, m$. Empirical Risk Minimization (ERM) is the task of searching a predictor that minimizes $L_s(h)$ that makes sense, since the only information available to the learner is just the training set S (that should be a snapshot of the world), moreover, since D and f are unknown, minimizing the true error is not possible.

ERM rule, however, may have the side effect that learns "too well" training data, that does not work well on new data from D. This effect is called *overfitting*, and a common solution to prevent it, is searching h in a restricted set of predictors. In this way, ERM rule can be written as:

$$hs \in argmin_{h \in H} L_s(h)$$

By restricting on a finite set of hypotheses H, we *bias* the learner, since we cannot guarantee that the algorithm picks the optimal one.

ERM rule depends on training set S, so a general assumption is to consider elements in S, independently and identically distributed according to the distribution D. However, picking a restricted number of samples, it is possible that those data are not *representative* of the distribution, and the learner may be affected by this issue. Furthermore, it is more realistic to not assume that labels are fully determined by features on input data. Saying δ the probability that the training set is not representative, and ϵ an *accuracy parameter*, it is possible to interpret $(1 - \delta)$ as a *confidence parameter*, and the event $L_{(D,f)}(h_s) > \epsilon$ a failure of the learner.

After these assumptions, the goal of the learner is to find a predictor that is probably approximately correct, formally an hypothesis h_s that with probability $> (1 - \delta)$ (that determines how confident is the solution), upper bounds the generalization error $L_{(D,f)}(h_s) < \epsilon$ (how well the predictor has learned). Since D and f are still not known, the learner tries to find the optimal hypothesis by minimizing the empirical risk.

Machine Learning Tasks

The model discussed so far can be applied in multiple machine learning tasks.

- Classification. It is the task of predicting a discrete and finite set of labels. Such algorithms have access to a set of correctly classified samples, and on the base of these instances, learns a predictor that taken a new instance, it predicts its own class. Classification ca be binary or multiclass, depending on the size of the possible set of labels. This work is focused on this task, more precisely on both binary and muticlass classification.

- Regression. It is the task of fining some simple patterns on data that a re functional dependencies beyween the sets X and Y.

A supervised learning task can be seen as an optimization problem that aims to minimize a *loss function*. Introducing some formalism, a loss function ℓ is a function from $X \times Y$ to the set of non negative real numbers

$$\ell : X \times Y \to \mathbb{R}_+$$

Saying z belonging to $X \times Y$, the generalization error becomes:

$$L_D(h) \overset{\text{def}}{=} \mathbb{E}_{z \sim D}[\ell(h, z)]$$

While the empirical risk becomes:

$$L_s(h) \overset{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} \ell(h, z_i)$$

The loss function ℓ is characteristic of the learning task: in binary and multiclass classification, the so called 0-1 loss is the correct choice:

$$\ell_{0-1}(h, (x, y)) \overset{\text{def}}{=} \begin{cases} 0 & \text{if} \quad h(x) = y \\ 1 & \text{if} \quad h(x) \neq y \end{cases}$$

That intuitively counts the number of misclassified samples. While in regression tasks, the squared loss is a better choice:

$$\ell_{sq}(h, (x, y)) \overset{\text{def}}{=} (h(x) - y)^2$$

2.3.3 Model Selection and Validation

Suppose we are facing a classification problem. As explained so far, the common approach is applying the ERM rule on a training set S restricting to a finite set of hypotheses H. However, we want to try different algorithms (or same algorithm with different parameters) on the same training set, and finally select the best predictor. In this way we need new data in order to evaluate the performance of the algorithms, so usually the approach is the following: the training set is splitted in two parts, training and validation sets. The first set is used to train the whole set of algorithms, obtaining the set $H = h_1, \ldots, h_r$ of best predictors, then these hypotheses are tested on the validation set, and only the hypothesis that reaches the

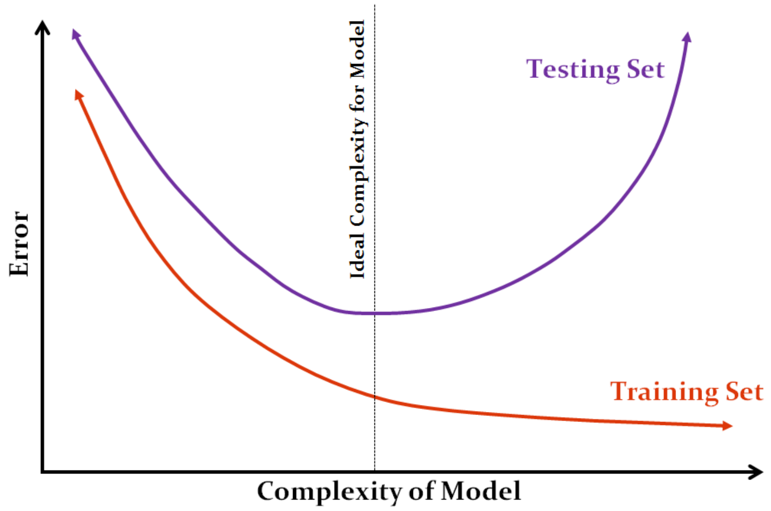

Figure 2.2: Learning curve for training and validation

minor loss is picked (actually we apply the ERM rule on H).

In Figure 2.2 it is shown the usual behavior in validation: training error is monotonically decreasing, because on growing the model complexity, the data are more well fitted. The validation error, instead, at some point starts increasing, that is symptom of overfitting. In this way, the predictor that should be chosen on validation is the one with complexity equal to the point on which the validation curve starts increasing, limiting both overfitting and underfitting.

In practical problems, the dataset is splitted into three subsets: the training set is used to train the learning algorithms, the validation set is used to select the best learned model, and the test set is used to estimate the true error. This last set cannot be used in the learning process, otherwise it can bias the final result, so it must be used only once. If the learning fails, new data must be used for test the final model.

K-Fold Cross Validation

Sometimes data are not plenty, so test-validation split may reduce too much the training set size, so a common approach is the so called *k-fold cross validation* [18]. In this technique, data is partitioned in k subsets of equal size called *folds*, usually respecting the labels distribution in a process called *stratification*. Then, for k iterations, the learning algorithm is trained with $k - 1$ folds, and validated using the last one, as shown in Figure 2.3. Finally, the score is calculated as the average of the k scores obtained with the k validation tests.

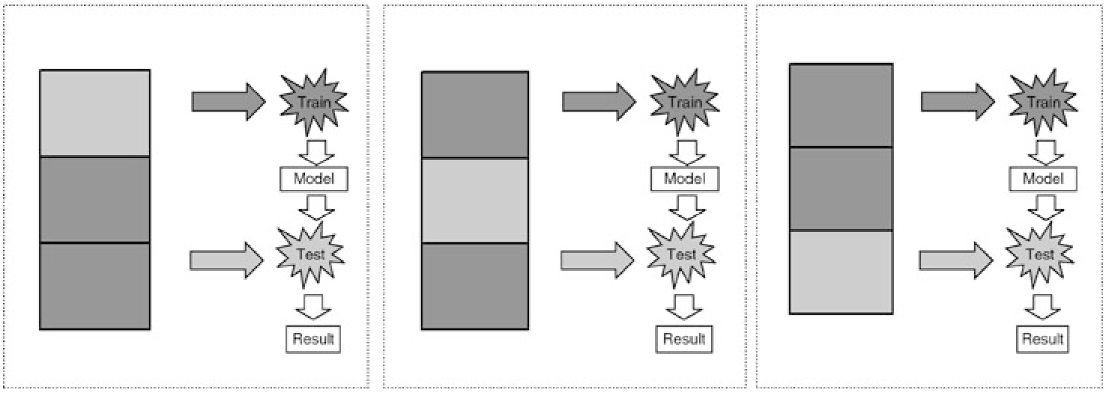

Figure 2.3: Example of k-fold cross validation with k=3

2.3.4 Linear Models

One of the most important and commonly used hypotheses class is the one of *linear predictors*. Since this class is used also in this work, it deserves a dedicated focus.
Define the class of affine functions as:

$$L_d = \{h_{\mathrm{w},b} : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

where

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \left(\sum_{i=1}^{d} w_i x_i \right) + b$$

L_d is a set of functions, parameterized by $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, that take in input a vector $x \in \mathbb{R}^d$ and outputs $\langle \mathrm{w}, \mathrm{x} \rangle + b$, which is a scalar. The different hypotheses classes are compositions of a function $\Phi : \mathbb{R} \to Y$ on L_d, for instance in binary classification the function Φ can be chosen as the sign function.
Usually the parameter b is incorporated into $\mathbf{w}$ adding one extra coordinate equal to 1 as follows:
let $w' = (b, w_1, \dots, w_d) \in \mathbb{R}^{d+1}$ and let $x' = (b, x_1, \dots, x_d) \in \mathbb{R}^{d+1}$.
Therefore,
$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \langle \mathbf{w}', \mathbf{x}' \rangle$ so in general the parameter b is omitted because included in $\mathbf{w}$. It follows that each affine function can be thought as a homogeneous linear function in an upper dimension, with a coordinate equal to 1.
The class used for binary classification is the *halfspaces class*. It is made by the composition

of the class of linear functions L_d with the sign function, so $X = \mathbb{R}^d$ and $Y = \{-1, 1\}$.

$$HS_d = sign \circ L_d = \{sign(h_{\mathbf{w},b}(\mathbf{x})) : \mathbf{x} \in \mathbb{R}^d, h_{\mathbf{w},b}(\mathbf{x}) \in L_d\}$$

Each hypothesis forms an hyperplane that is perpendicular to $\mathbf{w}$, and intersects the vertical axis in $(0, -\frac{b}{w_2})$. The instances that are ”above” the hyperplane, which means on the same direction of $\mathbf{w}$, are labeled positively, while the others negatively, as shown in Figure 2.4.

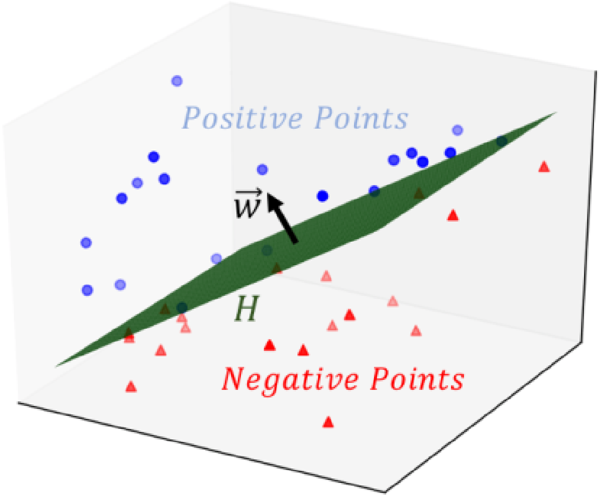

Figure 2.4: Halfspace in $\mathbb{R}^3$

2.3.5 Regularization

In common machine learning problems, it is important to be on guard against overfitting. In some cases a philosophical solution comes from the *Occam's Razor*, which states that from a pool of solutions, it is better to choose the simplest one, on the basis that they are capturing more fundamental aspects. *Regularization* is an application of this principle to ERM: instead of searching the model that best fits the training data, a penalty based on the model complexity is added. In fact, in general, algorithms tend to overfit the training data, learning a model that is more complex, so adding a penalty on the optimization function, makes the learner more *stable*, that also generalize better [19]. The ERM rule becomes:

$$argmin_{\mathbf{w}}(L_s(\mathbf{w}) + R(\mathbf{w}))$$

Where $R(\mathbf{w}))$ is a regularization function $R : \mathbb{R}^d \to \mathbb{R}$. This approach is called *Regularized Loss Minimization* (RLM). Intuitively, the model complexity is measured by the value of the regularization function, and the algorithm balances between low empirical risk and simpler hypotheses.
Some common regularization functions are the followings:

- L1 Regularization: the regularization function is

$$R(\mathrm{w}) = \lambda \sum_{i=1}^{m} |w_i|$$

- L2 Regularization: the regularization function is

$$R(\mathrm{w}) = \lambda \sum_{i=1}^{m} (w_i)^2$$

Writing the expected risk of a learning algorithm as

$$\mathbf{E}_S[L_D(A(S))] = \mathbf{E}_S[L_S(A(S))] + \mathbf{E}_S[L_D(A(S)) - L_S(A(S))]$$

it is shown that it is composed by two terms: the first one describes how well the algorithm fits the training data, and increases with λ, while the second one reflects just the difference between the true risk and the empirical risk. This second term can be seen as the stability of the learner, and decreases with λ. We are facing a tradeoff between fitting and stability that reflects on the choice of the parameter λ.

2.3.6 Gradient Descent

The goal of a machine learning task is to minimize the risk function $L_d(h)$. It is not possible to minimize it directly, since it depends on the unknown distribution D, so another approach is mandatory. In general there are multiple possible approaches, but the most used derives from the *Gradient Descent* (GD) [20].
Recall that the gradient $\nabla f(\mathbf{w})$ of a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is the vector $\nabla f(\mathbf{w}) = \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \dots, \frac{\partial f(\mathbf{w})}{\partial w_d}\right)$, that points to the direction of maximum growth of the function. Gradient descent is an iterative algorithm that starts from a initial point $\mathbf{w}^{(0)}$, and at each iteration t updates the point by ”moving” a step in the opposite direction of the

gradient, as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)})$$

where $\eta > 0$ is a parameter that intuitively defines the length of the step. After T iterations of the algorithm, the value of the function should converge to a value $(w)^{(T)}$, which is the supposed minimum (since it is an heuristic, the result is not guaranteed to be the absolute minimum, but just a local minimum). The final result can be either the last vector found $\mathbf{w}^{(T)}$, the best performing $argmin_{t \in T} f(\mathbf{w}^{(t)})$, or the average $\bar{\mathbf{w}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{w}^{(t)}$ (preferred in the case of non differentiable functions). The pseudocode of GD is shown in Algorithm 2.1.

Algorithm 2.1 Gradient Descent

```
parameters: scalar $\eta > 0$, integer $T > 0$
initialize: $\mathbf{w}^{(1)} = \mathbf{0}$
for $t = 1, 2, \ldots, T$
    $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)})$
output: $\bar{\mathbf{w}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{w}^{(t)}$
```

However, in Gradient Descent it is required to know the exact gradient of the function, which is unfeasible especially when working on large sets of data, so an approximation may be useful as well. *Stochastic Gradient Descent* (SGD) requires that the expected value of the update direction is equal to the gradient direction, which is a great advantage in terms of calculations, since it requires less points to be calculated.b The pseudocode of SGD is shown in Algorithm 2.2.

Algorithm 2.2 Stochastic Gradient Descent

```
parameters: scalar $\eta > 0$, integer $T > 0$
initialize: $\mathbf{w}^{(1)} = \mathbf{0}$
for $t = 1, 2, \ldots, T$
    choose $\mathbf{v_t}$ at random from a distribution such that $\mathbb{E}[\mathbf{v_t} | \mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)})$
    $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla \mathbf{v_t}$
output: $\bar{\mathbf{w}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{w}^{(t)}$
```

A visual demonstration of the behavior of both Gradient Descent and Stochastic Gradient Descent is shown in Figure 2.5. It is shown that SGD to reach the minimum, takes more steps in a direction that is not exactly the gradient direction, but only on the average.

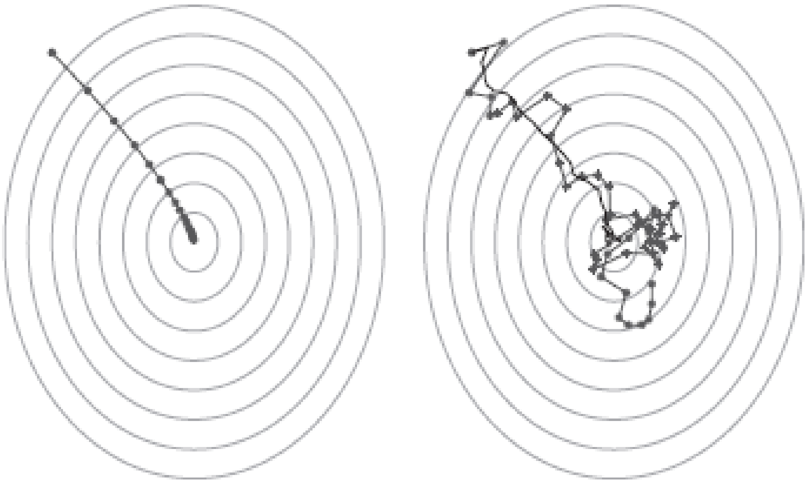

Figure 2.5: Behavior of Gradient Descent (left) and Sochastic Gradient Descent (right).

2.3.7 Feature Selection

As early explained, supervised learning is based on finding a proper function that joins the input vectors of features $\mathbf{X}$ with their label Y. Sometimes, the output Y does not depend by all features $(X_1, \ldots, X_d)$, instead, it is decided by a subset of them $(X_{(1)}, \ldots, X_{(r)})$, with $r < d$. With sufficient data and time, it is fine to use all input features for learning a classification algorithm (including also those irrelevant), but in practice there are motivations for that it is preferable to select only relevant ones.

- The irrelevant features will induce great computational cost.
- The irrelevant features may lead to overfitting, for instance if in sentiment analysis the name of the person for which it is going to classify the comment is included in the features set, then the classification algorithm may depend on this (which obviously is a mistake).
- Since the goal of supervised classification is to estimate a function with respect to the input features, it is reasonable to ignore features with low impact on the decision making, in order to keep the model as simple as possible.

The feature selection problem has been studied for many years, in fact many solutions are proposed for machine learning tasks. Below are presented some heuristic feature selection procedures that have been implemented in this work.

Feature Selection for Linear SVM

In linear classifiers, the outcome of a sample vector $x = (x_1, \ldots, x_d)$ is calculated as $y = sign(\mathbf{w}^T\mathbf{x})$. In [21] it is explained a feature selection method that uses the intuitive fact that weights with small absolute value $|w_j|$ does not contribute much on the decision ov the value y. This means that those features are not relevant on the final decision, so they could be discarded. This situation is valid also for SVM with linear kernel, while for other non-linear kernels it is not applicable.
The described strategy involves the linear SVM classifier, and it is the following:

- Train the classifier with training data;
- Eliminate the features whose weight is close to zero, formally setting a threshold T, just keep the features $\{w_i : i \in [d], |w_i| > T\}$;
- Finally, re-train the model using just the selected features.

Feature Selection L_1 norm Regularization

In regularization, it is optimize a function composed by both empirical risk, and a regularization function. In case of L_1 norm regularization, the term

$$\lambda \sum_{i=1}^{d} |w_i|$$

especially when λ is large, forces some of the weights w_i to be exactly equal to zero [22]. A typical behavior of the values of the weights with L_1 norm regularization is shown in Figure 2.6.

It is possible to see that L_1 norm regularization selects automatically some weights to be nonzero, while sets zero the others. In this situation, the features to be kept are obviously the one with nonzero values. The strategy is the same as previously mentioned:

- Train the classifier with training data;
- Eliminate the features whose weight is zero (or close to);
- Finally, re-train the model using just the selected features.

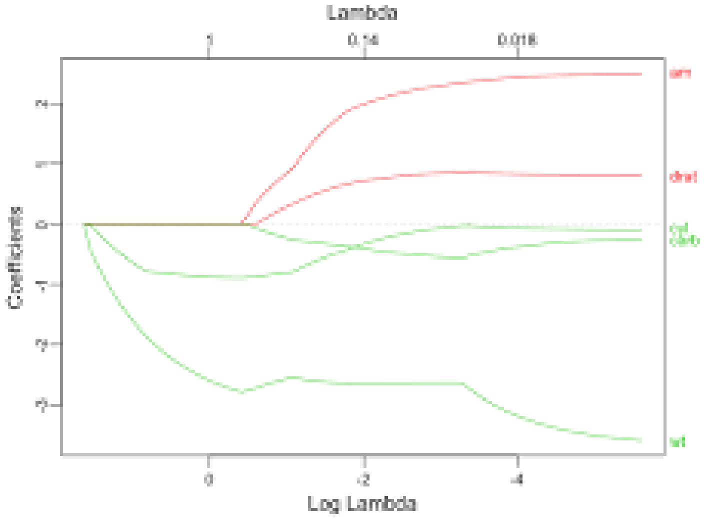

Figure 2.6: Behavior of weights on L_1 norm regularization on with respect to λ.

Information Gain Feature Selection

Intuitively, *Information Gain* (IG) measures the amount of information in bits about the class prediction, if the only information available is the presence of a feature and the corresponding class distribution [23].

Given a random variable X, the following definitions comes from Shannon's Information theory [24]:

- The Information associated to an event x from the distribution X is

$$I(x) = \log_2 \frac{1}{\mathbb{P}_X(x)} = -\log_2 \mathbb{P}_X(x)$$

- The Entropy of a random variable X is the average amount of information of the possible events

$$H(X) = -\sum_{x \in X} \mathbb{P}_X(x) \log_2 \mathbb{P}_X(x)$$

Both Information and Entropy are pure numbers, but "measured" in *bits*.

Concretely, information gain measures the expected reduction in entropy by choosing a given feature. The information gain of a training set S with respect to a feature $i \in [d]$

is calculated as:

$$IG(S,i) = H(S) - \sum_{v \in values(X_i)} \mathbb{P}(X_i = v) H(S_{X_i=v})$$

Where X_i is the set of values of the feature i, and $S_{X_i=v}$ is the training set restricted to the samples where $X_i = v$.
Feature selection using information gain works as follows: given the training dataset S, the information gain score is calculated for each feature X_i. Next, all scores are sorted in a ranking, then the most important features are selected. The selection can be done either by choosing a threshold on the information gain, or by choosing the size of the set of the selected features.

In the followings sections are presented some of the most used algorithms for binary classification, that are also exploited in this work.

2.3.8 Logistic Regression

Logistic Regression is a model that analyzes the relationships between multiple independent variables, and a categorical dependent variable, namely between the features and the labels in case of a supervised classification task [25].
In the context of linear models, it was described the class of hyperplanes, which is a set of functions from $\mathbb{R}^d$ to $\{0,1\}$. However, in some cases it may be useful to have a measure of confidence about the prediction. Logistic Regression provides that information by defining the probability that the label of an input instance $\mathbf{x}$ is 1. Calling $\mathbb{P}_{\mathbf{w}}[Y|X]$ that conditional probability to get the label Y given the instance X, under the estimator with parameter $\mathbf{w}$. Statistical theory suggest that the probabilistic classifier should be learned from the training set S using the MLE estimator as:

$$\hat{\mathbf{w}}_{MSE} = argmax_{\mathbf{w}} \quad \mathbb{P}_{\mathbf{w}}[y_1|x_1], \ldots, \mathbb{P}_{\mathbf{w}}[y_n|x_n]$$

In Logistic Regression the conditional probability $\mathbb{P}_{\mathbf{w}}[Y|X]$ is expressed as:

$$\mathbb{P}_{\mathbf{w}}[Y = y|X = x] = \frac{1}{1 + e^{y\langle \mathbf{w}, \mathbf{x} \rangle}}$$

where y can be either -1 or 1. This result is obtained by the composition of the prediction of the classifier $y = \langle \mathbf{w}, \mathbf{x} \rangle$ (which has no bounds on the values) with the sigmoid function

$$sig(t) = \frac{1}{1+e^{-t}}$$

which is shown in Figure 2.7.

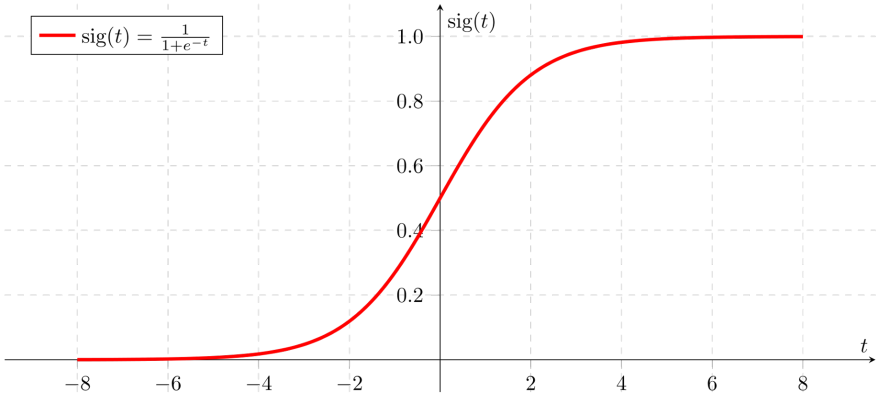

Figure 2.7: Sigmoid function.

The hypotheses class is therefore

$$H_{logreg} = sig \circ L_d = \{sig(\langle \mathbf{w}, \mathbf{x} \rangle) : \mathbf{x}, \mathbf{w} \in \mathbb{R}^d\}$$

An interpretation of the formulas may be useful to understand. As long the value $\langle \mathbf{w}, \mathbf{x} \rangle$ is large, the predictor can be considered sure about the prediction (the value is far from the separating hyperplane). Contrary, if the value $\langle \mathbf{w}, \mathbf{x} \rangle$ is low, it means that $\mathbf{x}$ falls nearby the separating hyperplane, so the prediction may be less confident. Applying the prediction value to the sigmoid function, as long the value is large, the result is getting closer to 1, while when the prediction value becomes negative, the final value is getting closer to 0. Therefore, the value obtained by the sigmoid can be viewed as the probability that the input value $\mathbf{x}$ has label 1.

Summarizing all the concepts, the final goal is to have the value $y\langle \mathbf{w}, \mathbf{x} \rangle > 0$ for each $(x, y) \in S$ (or more conveniently $\gg 0$). Recalling the Maximum Likelyhood Estimator solution as

$$\begin{aligned}\hat{\mathbf{w}}_{MSE} &= argmax_{\mathbf{w}\in\mathbb{R}^d} \quad \mathbb{P}_{\mathbf{w}}[Y|X] \\ &= argmax_{\mathbf{w}\in\mathbb{R}^d} \quad \prod_{i=1}^{n} \frac{1}{1+e^{-y_i\langle\mathbf{w},\mathbf{x_i}\rangle}} \\ &= argmin_{\mathbf{w}\in\mathbb{R}^d} \quad \sum_{i=1}^{n} log(1+e^{-y_i\langle\mathbf{w},\mathbf{x_i}\rangle})\end{aligned}$$

which derives the final optimization problem as:

$$argmin_{\mathbf{w}\in\mathbb{R}^d} \quad \sum_{i=1}^{n} log(1+e^{-y_i\langle\mathbf{w},\mathbf{x_i}\rangle})$$

The same result can be reached following the ERM rule by setting the loss function increasing with the probability that the estimator is wrong, formally defining

$$\ell(h_{\mathbf{w}},(\mathbf{x},y)) = log(1+e^{-y\langle\mathbf{w},\mathbf{x}\rangle})$$

The associated ERM problem becomes

$$argmin_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{n}\sum_{i=1}^{n} log(1+e^{-y_i\langle\mathbf{w},\mathbf{x_i}\rangle})$$

2.3.9 SVM

In context of binary classification, *Support Vector Machines* became one of the most popular classification methods [26]. SVM is based on the hypotheses class of hyperspaces, already described, and the main concept is to find the hyperplane that best classifies the training data.

Hard-SVM

Suppose that training data are linearly separable, formally

$$\forall i \in [m], y_i(\langle\mathbf{w},\mathbf{x}_i\rangle) > 0$$

intuitively, there exists an hyperplane that classifies correctly every instance ot the training set, like the one shown in Figure 2.8

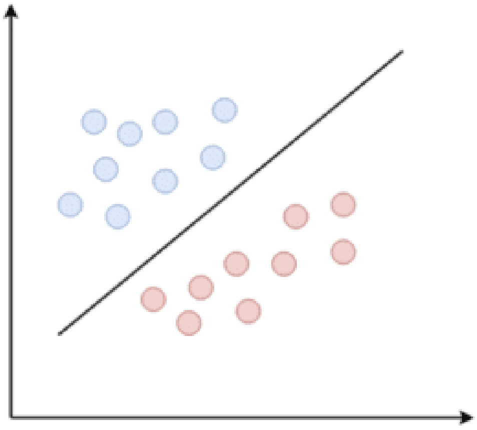

Figure 2.8: Example of linearly separable set.

Given a set of possible separating hyperplanes, it is intuitively better to prefer the one with largest *margin*. The margin of an hyperplane with respect to a training set is defined to be the minimal distance between the hyperplane and a point of the set, formally

$$min_{i\in[m]}|\langle \mathbf{w}, \mathbf{x}_i\rangle + b|$$

Therefore, the Hard-SVM rule, which finds the largest margin in linearly separable case is:

$$argmax_{(\mathbf{w},b):\|\mathbf{w}\|=1} min_{i\in[m]} \quad |\langle \mathbf{w}, \mathbf{x}_i\rangle + b| \quad s.t. \quad \forall i \in [m], y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle) > 0$$

Or another equivalent formulation as a quadratic optimization problem is

$$(\mathbf{w}_0, b_0) = argmin_{(\mathbf{w},b)} \|\mathbf{w}\|^2 \quad s.t. \quad \forall i \in [m], y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle) \geq 1$$

The output should be $\hat{\mathbf{w}} = \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|}, \hat{b} = \frac{b_0}{\|\mathbf{w}_0\|}$.

The result of the Hard-SVM optimization problem is shown in Figure 2.9.

The points at minimum distance from the hyperplane are called *Support Vectors*.

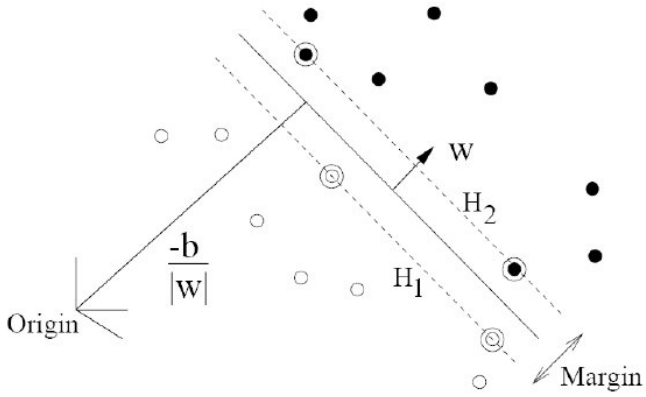

Figure 2.9: Hard-SVM solution.

Soft-SVM

The Hard-SVM formulation supposes that data are linearly separable, however in usual cases, data are not. To handle this situation, Soft-SVM model permits to some points to fall in the wrong side of the hyperplane, but adding a penalty on the objective function represented by *slack* variables ξ_i, that control how far from the wrong side the point $\mathbf{x}_i$ is. The optimization problem becomes

$$argmin_{(\mathbf{w},b)}\frac{1}{2}\left\|\mathbf{w}\right\|^2 + C\sum_{i=1}^{m}\xi_i \quad s.t. \quad \forall i \in [m], y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) \geq 1 - \xi_i, \xi \geq 0$$

The parameter C is a regularization parameter, that controls the trade-off between margin's size and the training errors.

Solving both the Hard-SVM and Soft-SVM optimization problems is quite hard, due to the complicated constraints [27]. The mathematical tool for simplify the problem is the Lagrangian duality theory [28].

From theory, calling $\mathbf{w}_0$ the solution of Hard-SVM problem, and calling $I = i : |\langle \mathbf{w}_0, \mathbf{x}_i \rangle| = 1$ (the set of indices that defines the support vectors), then there exist coefficients $\alpha_1, \ldots, \alpha_m$ such that

$$\mathbf{w}_0 = \sum_{i \in [I]} \alpha_i \mathbf{x}_i$$

The same result is valid also for Soft-SVM problem. The Lagrangian duality theory leads to

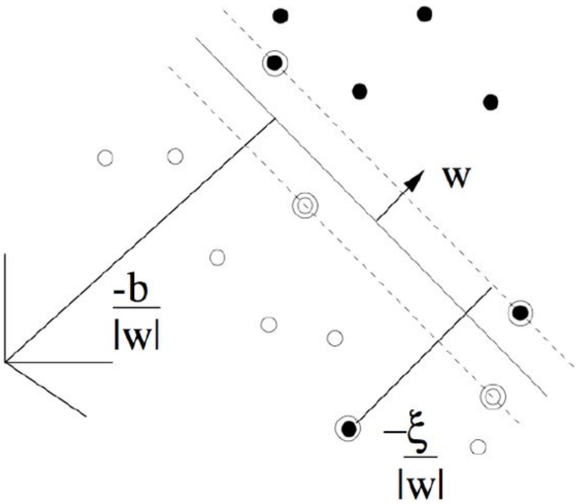

Figure 2.10: Soft-SVM solution.

solving the Hard-SVM dual optimization problem

$$max_{\alpha \in \mathbb{R}^m} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right)$$

subject to

$$\forall i, \quad \alpha_i \geq 0, \quad \sum_i y_i \alpha_i = 0$$

and the Soft-SVM dual optimization problem

$$max_{\alpha \in \mathbb{R}^m} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right)$$

subject to

$$\forall i, \quad 0 \leq \alpha_i \leq C, \quad \sum_i y_i \alpha_i = 0$$

The resulted dual problems are much simpler, since the constraints are less complicate.

Kernels

Sometimes points in the training set are far from being linearly separable, like the one in Figure 2.11. This is because the expressiveness of the halfspaces class is limited. However, it is possible to map the original instance space into another space (possibly of a higher dimension), and then learn a halfspace in that space.

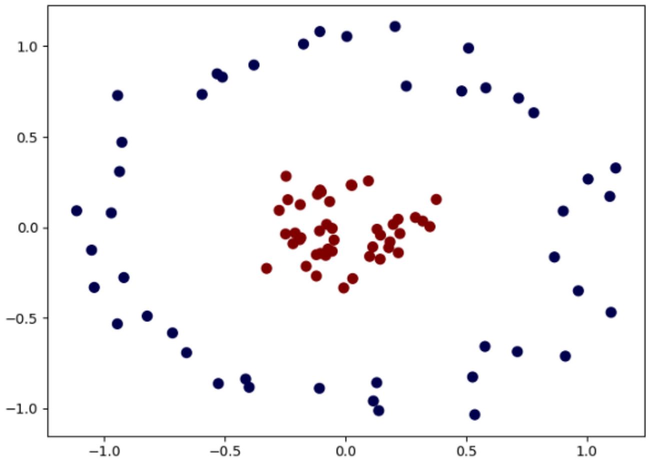

Figure 2.11: Example of non separable data.

The basic paradigm become:

- Given a domain X, choose a mapping $\phi : X \to F$, for some feature space F;
- Given a training set $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$, apply the mapping to the instances $\hat{S} = ((\phi(\mathbf{x}_1, y_1)), \ldots, (\phi(\mathbf{x}_m), y_m))$;
- Train a linear predictor h on $\hat{S}$;
- New predictions of some point $\mathbf{x}$ are made on $h(\phi(\mathbf{x}))$.

An example of solution with kernel is shown in Figure 2.12, where data are mapped with the function

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3$$

$$\mathbf{x} \to (\mathbf{x}, \|\mathbf{x}\|)^2$$

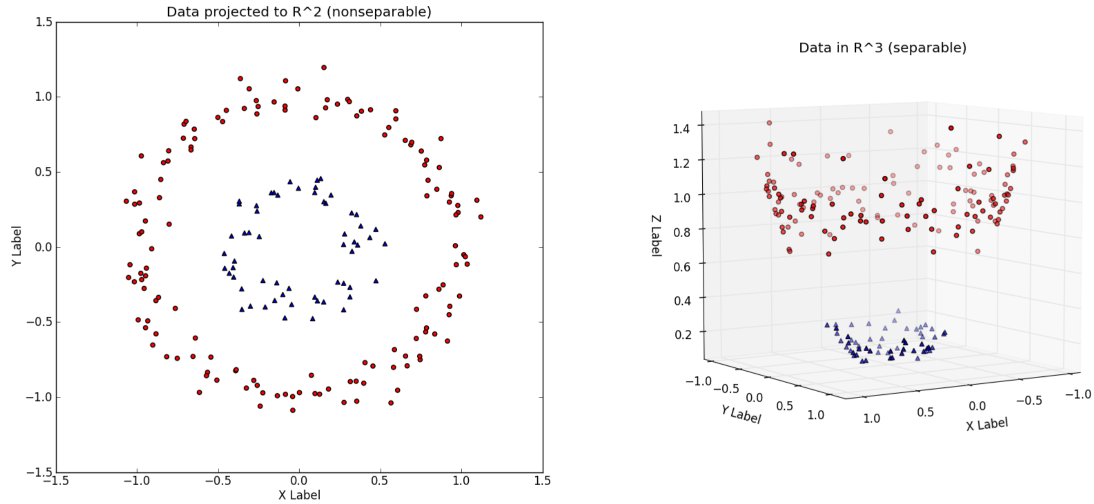

Figure 2.12: Exaxmple of solution with kernel.

Embedding the input space into a higher dimensional space, makes the halfspace class more expressive. However, computing the separating hyperplane in a high dimension may be computational expensive. Looking at the SVM dual optimization problem, the increasing computational cost comes from the calculation of $\langle \mathbf{x}, \mathbf{x}' \rangle$, to the calculation of $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ (called also *Kernel Function* $K(\mathbf{x}, \mathbf{x}')$). However, for some kernel functions, it is possible to calculate directly $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ without passing through firstly the calculation of the mapping and then the inner product, that saves some computation. This procedure is called *Kernel Trick*.

Some examples of the most used kernel functions are:

- K-degree Polinomial kernel

$$K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k$$

- Gaussian kernel

$$K(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma}}$$

- Radial Basis Function (RBF) kernel

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma(\mathbf{x}-\mathbf{x}')^2}$$

Where σ on gaussian kernel, and γ on RBF kernel are parameter. is shown in Algorithm ??

2.3.10 Random Forest

Recalling that the goal of classification is to minimize a cost function $L_D(h)$ by optimizing the empirical risk $L_s(h)$ (eventually including the regularization term), the so called *Ensemble* models try to obtain the optimal value from a collection of multiple *base learners* $h_1(\mathbf{x}), \ldots, h_J(\mathbf{x})$. These learners are then combined to obtain the final prediction $h(\mathbf{x})$ by "voting" the most common outcome.

$$h(x) = argmax_{y \in Y}|[j \ for \ j \in [J] \ if \ h_j(\mathbf{x}) = y]|$$

Random Forest is an ensemble learning model, based on *Decision Trees* base learners [29].

Decision Trees

A decision tree is a flowchart-like tree structure, where each internal node represent a test on an attribute, each branch represent an outcome of the test, class label is represented by each leaf node [30]. For instance, suppose we want to predict whether a person is fit, given some simple information like the age, or what he eat. A decision tree that makes that classification may be the one in Figure 2.13.

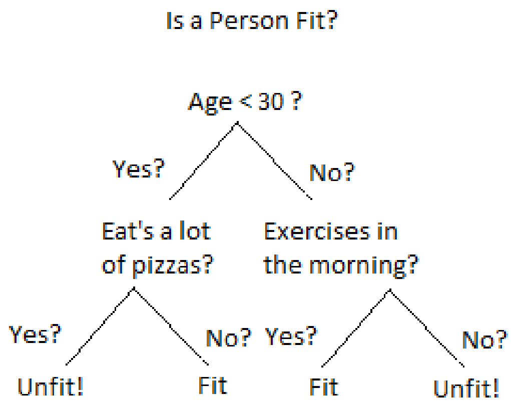

Figure 2.13: Decision Tree for decide whether a person is fit.

The task of learning a decision tree consists on determine the splits that define the tree, such that the loss is minimized. Searching for the optimal decision tree that minimizes the loss is computationally hard, so it is preferable to adopt heuristic rules where the tree is built

step-by-step by searching a locally optimal solution.
A general procedure for building a decision tree starts with a single leaf tree (root), and assigns this node the label according to the majority vote among all labels on the training set. Now, for a series of iterations, it is examined the effect of splitting a single leaf. Defining as *gain* a measure of improvement due to the split, the goal for each step is to find the split that reaches the maximum gain.
A possible implementation of decision tree learning algorithm, based on information gain metrics for the splitting criterion, is known as ”ID3” (Iterative Dichotomizer 3) [31].

At each iteration, the algorithm selects the attribute with highest information gain as splitting attribute. When information gain approaches to zero, the growing of the tree stops. A recursive solution is shown in Algorithm 2.3.

Algorithm 2.3 ID3(S, A)

```
input: training set $S$, feature subset $A \subseteq [d]$
if all examples in $S$ are labeled with $1$
    return a leaf $1$
if all examples in $S$ are labeled with $0$
    return a leaf $0$
if $A = \emptyset$
    return a leaf whose value = majority of labels in $S$
else
    Let $j = argmax_{i \in A} IG(S, i)$
    if all examples in $S$ have the same label
        return a leaf whose value = value of the majority of the labels in $S$
    else
        Let $T_1$ be the tree returned by ID3($\{(\mathbf{x}, y) \in S : S : x_j = 1\}, A\ j)$
        Let $T_2$ be the tree returned by ID3($\{(\mathbf{x}, y) \in S : S : x_j = 0\}, A\ j)$
        return left child of the input node = T2, right child of the input node = T1
```

A similar approach described in ”CART” (Classification and Regression Trees) [32], involves the *Gini* index, instead of information gain. Gini index is calculated as

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Where p_i is the probability of a value i in the distribution of the interested attribute. The Gini index is calculated for each attribute, and the split is made with respect to the attribute with lowest value of Gini index.

Random Forest

Decision Trees lead easily to overfitting. A solution to prevent the issue, is to construct an ensemble of trees, which is the basic of Random Forest Classifiers. In Rando Forest, each decision tree is constructed by applying the mentioned algorithm on the training set S and an additional randome vector Θ, where Θ is sampled from some distribution. The final prediction is obtained by the majority of the votes. There are many ways to define the vector Θ, a possible one is the following:

- Take a random subsample of S, obtaining S';
- Construct a sequence $I_1, I_2, \ldots$ where each I_t is a subset of $[d]$ of size k, generated by sampling k random elements from $[d]$;

Finally, each Decision Tree is based on the set S', where at each split, it is restricted to choosing among the set of features I_t

2.3.11 Naïve Bayes

Naïve Bayes is a subset of Bayesian decision theory [33]. It is called naïve because the formulation makes some assumptions. The classifier assumes that each feature only depends on the class of the instance (that implies also that all features are independent each other), as shown in Figure 2.14. However, in real world cases, it is not possible to guarantee this assumption, but the classifier works still reaches good results.

Let y denote the class of an observation $\mathbf{x}$. Predict the class of $\mathbf{x}$ by searching the class y that maximize the value of the Bayes Rule (posterior probability)

$$\mathbb{P}(y|\mathbf{x}) = \frac{\mathbb{P}(y)\mathbb{P}(\mathbf{x}|y)}{\mathbb{P}(\mathbf{x})}$$

Exploiting the fact that the features $x_1, \ldots, x_d$ are independent, the following equation is sufficient to predict the most probable class

$$\mathbb{P}(y|\mathbf{x}) = \frac{\mathbb{P}(y)\prod_{i=1}^{d}\mathbb{P}(x_i|y)}{\mathbb{P}(\mathbf{x})}$$

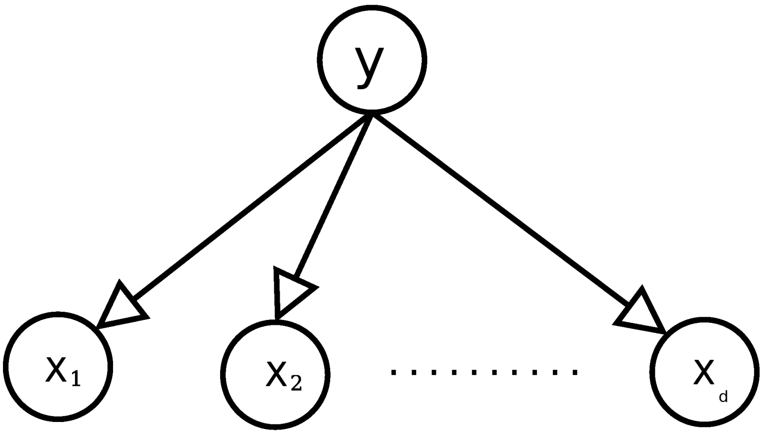

Figure 2.14: Features independency on Naïve Bayes classifier.

2.3.12 Multiclass Classification

2.3.13 Classification Metrics

2.4 Sentiment Analysis Algorithms

2.4.1 BPEF

This is some random quote to start off the chapter.

Firstname lastname

3 Dataset

Nulla facilisi. In vel sem. Morbi id urna in diam dignissim feugiat. Proin molestie tortor eu velit. Aliquam erat volutpat. Nullam ultrices, diam tempus vulputate egestas, eros pede varius leo.

Quoteauthor Lastname

4 Algorithms

Nulla facilisi. In vel sem. Morbi id urna in diam dignissim feugiat. Proin molestie tortor eu velit. Aliquam erat volutpat. Nullam ultrices, diam tempus vulputate egestas, eros pede varius leo.

Quoteauthor Lastname

5 Experiments

6
Conclusions and Future Works

References

[1] D. Zimbra, A. Abbasi, D. Zeng, and H. Chen, "The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation," *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 2, pp. 5:1–5:29, Aug. 2018. [Online]. Available: http://doi.acm.org/10.1145/3185045

[2] A. Bermingham and A. Smeaton, "On using twitter to monitor political sentiment and predict election results," in *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011).* Chiang Mai, Thailand: Asian Federation of Natural Language Processing, Nov. 2011, pp. 2–10. [Online]. Available: https://www.aclweb.org/anthology/W11-3702

[3] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, 2011. [Online]. Available: http://scholar.google.de/scholar.bib?q=info:7jxZLbM1mzwJ:scholar.google.com/&output=citation&hl=de&as_sdt=0&ct=citation&cd=54

[4] H. Rui, Y. Liu, and A. Whinston, "Whose and what chatter matters? the effect of tweets on movie sales," *Decis. Support Syst.*, vol. 55, no. 4, pp. 863–870, Nov. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.dss.2012.12.022

[5] F. Jurado and P. Rodriguez, "Sentiment analysis in monitoring software development processes: An exploratory case study on github's project issues," *Journal of Systems and Software*, vol. 104, pp. 82 – 89, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215000485

[6] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 1555–1565. [Online]. Available: https://www.aclweb.org/anthology/P14-1146

[7] A. Montejo-Ráez, E. Martínez-Cámara, M. Martín-Valdivia, and L. López, “Ranked wordnet graph for sentiment polarity classification in twitter,” *Computer Speech & Language*, vol. 28, 01 2013.

[8] K. Ravi and V. Ravi, “A survey on opinion mining and sentiment analysis: Tasks, approaches and applications,” *Knowledge-Based Systems*, vol. 89, pp. 14 – 46, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705115002336

[9] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[10] K. Anbananthen, J. Krishnan, M. Shohel Sayeed, and P. Muniapan, “Comparison of stochastic and rule-based pos tagging on malay online text,” *American Journal of Applied Sciences*, vol. 14, pp. 843–851, 09 2017.

[11] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown, “Text classification algorithms: A survey,” *CoRR*, vol. abs/1904.08067, 2019. [Online]. Available: http://arxiv.org/abs/1904.08067

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[13] V. Saifee and T. Jay, “Applications and challenges for sentiment analysis: A survey,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, 2013.

[14] F. Grässer, S. Kallumadi, H. Malberg, and S. Zaunseder, “Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning,” in *Proceedings of the 2018 International Conference on Digital Health*, ser. DH ’18. New York, NY, USA: ACM, 2018, pp. 121–125. [Online]. Available: http://doi.acm.org/10.1145/3194658.3194677

[15] A. Rane and A. Kumar, "Sentiment classification system of twitter data for us airline service analysis," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 01, July 2018, pp. 769–773.

[16] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[17] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. New York, NY, USA: Cambridge University Press, 2014.

[18] P. Refaeilzadeh, L. Tang, and H. Liu, *Cross-Validation*. Boston, MA: Springer US, 2009, pp. 532–538. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_565

[19] O. Bousquet and A. Elisseeff, "Stability and generalization," *Journal of Machine Learning Research*, vol. 2, pp. 499–526, 06 2002.

[20] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[21] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic, "Feature selection using support vector machines," 2002.

[22] J. Pereira, M. Basto, and A. Ferreira-da Silva, "The logistic lasso and ridge regression in predicting corporate failure," *Procedia Economics and Finance*, vol. 39, pp. 634–641, 12 2016.

[23] D. Roobaert, G. Karakoulas, and N. V. Chawla, "Chapter 22 information gain, correlation and support vector machines."

[24] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 7 1948. [Online]. Available: https://ieeexplore.ieee.org/document/6773024/

[25] H.-A. Park, "An introduction to logistic regression: From basic concepts to interpretation with particular attention to nursing domain," *Journal of Korean Academy of Nursing*, vol. 43, pp. 154–164, 04 2013.

[26] D. Fradkin and I. Muchnik, "Support vector machines for classification," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 01 2006.

[27] L. Bottou and C. jen Lin, "Support vector machine solvers," 2006.

[28] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*, 1st ed. Athena Scientific, 1996.

[29] A. Cutler, D. Cutler, and J. Stevens, *Random Forests*, 01 2011, vol. 45, pp. 157–176.

[30] H. Sharma and S. Kumar, "A survey on decision tree algorithms of classification in data mining," *International Journal of Science and Research (IJSR)*, vol. 5, 04 2016.

[31] R. Bedi, "Review of decision tree data mining algorithms : Id 3 and c4.5," 07 2015.

[32] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984, new edition [?]?

[33] S. Taheri and M. Mammadov, "Learning the naive bayes classifier with optimization models," *Int. J. Appl. Math. Comput. Sci.*, vol. 23, no. 4, pp. 787–795, Dec. 2013. [Online]. Available: https://doi.org/10.2478/amcs-2013-0059

Acknowledgments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetuer. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.