Sofia University "St. Kliment Ohridski",
Faculty of Mathematics and Informatics

Learning From Multiple Sources of Data

Gergana Angelova Lazarova

THESIS

for conferring of academic and scientific degree
DOCTOR
in professional field 4.6 Informatics and Computer Science
scientific speciality 01.01.12 Informatics (Artificial Intelligence)

Scientific supervisor:
Prof. Ivan Koychev

Sofia, 2015

TABLE OF CONTENTS

Structure

The thesis comprises of seven parts.

Part 1 is an introduction. It states the problem and introduces semi-supervised multi-view learning. It defines the aim of the thesis, as well as, the problems that should be tackled in order to achieve the aim. This part also presents the motivation for the thesis.

Part 2 is an overview of the field of semi-supervised learning. It focuses on semi-supervised multi-view learning. It also describes genetic algorithms.

Part 3 presents a novel approach to semi-supervised multi-view learning - *the semi-supervised multi-view genetic algorithm*. The proposed algorithm is compared to its supervised equivalent and experimental results are presented.

Part 4 introduces a learning system for sentiment analysis in Bulgarian. This system is based on the approach proposed in part 3. The development cycle of the system is described - the phases of data understanding, data acquisition, data translation, feature extraction and modelling. The architecture of the learning system concerning the software implementation is described. The performance of the system is evaluated and compared to the performance of a supervised one.

Part 5 presents a new approach to semi-supervised multi-view learning - *the multi-view teaching algorithm* – a modification of the standard co-training algorithm. The proposed algorithm is compared to its supervised equivalent and experimental results are presented.

Part 6 introduces an image segmentation system, which uses two sources of information. This system is based on the approach proposed in part 5. The architecture of the learning system concerning the software implementation is described. The performance of the system is evaluated and compared to the performance of a supervised one.

Part 7 is a conclusion. It states the scientific and applied contributions of the thesis and lists the publications and citations referred to in this work.

Acknowledgements

I thank my family, who were always supportive and motivating, especially my sister - Stanislava Lazarova, who was always there for me – throughout publication writing and conferences, giving me the energy and strength to finish my work. I would also like to thank my teachers from Sofia University - Prof. Ivan Koychev, Prof. Antoniy Popov, and everyone else I have worked with throughout the years of my education. I am also grateful to my colleagues from the Department of Software Technologies at the Faculty of Mathematics and Informatics of Sofia University, for their scientific passion and professionalism.

Благодаря Ви!

Thank You!

どうもありがとうございます！

1. Introduction

1.1 Problem Formulation

The human being is an intelligent agent, who can think and act rationally. A person is a complex biological system, comprised of various sensors, physical and chemical components. They can feel emotions, express love, happiness and sadness; they possess the unique ability to process data and extract useful information from it.

Since World War II, there has been considerable research in the field of artificial intelligence. This discipline studies the creation of computer software that is capable of intelligent behavior. It strives after better and better autonomous agents and learning systems, which are intelligent and can predict future events. These agents learn how to cope with everyday problems and how to react to the surrounding world. Still, one question remains open - can we create an intelligent agent "who" can not only act like a human, but can actually think and has a mind in exactly the same sense human beings do [1]?

Semi-supervised learning [2] is a state-of-the-art discipline of artificial intelligence, which is responsible for training an agent when only few labeled examples exist. There is not enough past history the agent can learn from. In this field, unlabeled instances are also used for training. The aim is to extract relevant information from the unlabeled instances, so that a good prediction algorithm can be learned.

When not enough labeled training instances are available, multiple sources of information are also explored. Such data sources will be referred to as *views* in the present work. Each example is represented by a set of characteristics in the various views. Semi-supervised multi-view learning uses labeled and unlabeled examples, as well as the agreement among multiple views in order to learn a good prediction algorithm.

The following dissertation proposes new approaches to semi-supervised multi-view learning, which are further applied to two domains – sentiment analysis and image segmentation.

1.2 Motivation

Why should unlabeled examples and multiple sources be used?

There are areas of research, where only few labeled examples exist. In such areas, labeling an example turns out to be a time-consuming process, requiring extra knowledge and skilled expertise in the field. Sometimes the process of classifying a new instance is not commercially beneficial or too expensive. Furthermore, there might not be trained experts in the field. In such cases, using unlabeled examples and multiple sources of information gives us further information, which will be useful for building a good machine learning algorithm.

1.3 Problem Significance and Present Interest

Since few years ago, this area of research was purely scientific and enthusiastic researchers were after the scientific contributions. With the booming development of the Internet and natural language processing, data became easier to access. Nowadays, the more commercial an application is, the more "important" for the society it is. Various systems appeared – most of them are web-sites offering different personal services. These newly-released systems suffer from labeled instances shortage. However, having only few labeled examples, how are preferences of users modeled? How can articles be recommended based on only few rated items? How to forecast future performance and results having a small amount of existing labeled instances?

The importance of researching this field lies in the fact that using *multiple sources of information and unlabeled instances* improves the classification accuracy of the algorithm. One percent improvement of the prediction can mean one percent increase in turnover.

Sentiment analysis is a modern discipline, which unites researchers from all over the world. There are so many published papers and implemented systems based on the English language. However, there has been little research and little labeled data in Bulgarian. Manually labeling examples can take hundreds of hours. Furthermore, sentiment analysis is generally subjective - two people alone might not agree on the sentiment of the example. This motivated the development of a novel learning system for semi-supervised multi-view sentiment analysis in the Bulgarian language. Since there are not enough classified examples in Bulgarian, which could be used for the training of a good classifier, a semi-supervised method was also explored. A second view is also required. As there are so many labeled examples in English, the following thesis focuses on using an English view in order to reinforce the Bulgarian one. Labeled examples in English are translated into Bulgarian by Google translate and unlabeled, i.e native Bulgarian examples, are translated into English.

In natural language processing (NLP) we can extract hundreds of thousands of features (curse of dimensionality). In such cases, when multi-view learning is applicable, it is reasonable to use multiple simultaneously running learners instead of one with too many features. It is especially applicable in algorithms, whose learning phase depends on the number of input characteristics (Neural Networks, Support Vector Machines).

Computer vision has been a major research spot for many years, including various subdivisions - image acquisition, preprocessing, feature extraction, segmentation techniques. It is a valuable discipline of research for the computer gaming industry, the movie industry, etc. This dissertation also focuses on this hot research area, on semi-supervised multi-view image segmentation in particular. In this field, a teacher labels few pixels beforehand and the rest of the pixels are used as unlabeled. Two sources of information are explored – the coordinates of the pixels and the color features (RGB values) of the pixels.

1.4 Thesis Aim

The aim of this thesis is the development of new approaches to multi-view semi-supervised learning.

1.5 Thesis Objectives

In order to meet the aim of the thesis, the following objectives are defined:

- Overview presentation of the field of semi-supervised learning and genetic algorithms;
- development of a new approach to semi-supervised multi-view learning. A *semi-supervised multi-view genetic algorithm* is developed and used for error minimization (the common error of multiple views is minimized);
- evaluation of *the semi-supervised multi-view genetic algorithm* and comparison to a supervised genetic algorithm;
- application of the semi-supervised multi-view genetic algorithm to a semi-supervised sentiment analysis system in Bulgarian (SASBG);
    - development of a distributed software architecture of SASBG, which can process big data;
    - evaluation of SASBG;
- development of a new approach (*multi-view teaching algorithm*). The multi-view teaching algorithm is a modification of the original co-training algorithm, in which one of the learners is weaker than the other;
- evaluation of *the multi-view teaching algorithm* and comparison to a supervised classifier;
- application of the multi-view teaching algorithm to a semi-supervised image segmentation system - SSISS;
    - development of a software architecture of SSISS;
    - evaluation of SSISS.

2. Theoretical Background and Overview of Semi-supervised Multi-view Learning

The following dissertation is in the field of multi-view semi-supervised machine learning. The theoretical background includes existing methods of supervised and unsupervised algorithms, which are carried out to create such in the field of multi-view semi-supervised learning. An overview of genetic algorithms is also presented as they are used to create a new approach to semi-supervised multi-view learning in part 3.

An expert in a specific domain (a "teacher") possesses knowledge about the field and the ability to classify examples of data. Based on whether an expert has classified an instance or not, there are two types of examples: *labeled* and *unlabeled* ones.

Definition 1. *Label*. A label *y* is the desired prediction on an instance *x*.

Definition 2. a *labeled example*: Let $\mathrm{X} = (\mathrm{X}_1,\ldots,\mathrm{X}_\mathrm{d},\mathrm{Y})$ be a (*d+1*)-dimensional vector, corresponding to the features of the instances and the value of the classification/regression function.

Let the set of labeled examples be:

$$D_1 = \{(x_i, y_i)\}_{i=1}^{l}.$$

l is the number of labeled examples. For all x_i, the corresponding values of the classification/regression function y_i have been defined.

Definition 3. *an unlabeled example* - Let $\mathrm{X} = (\mathrm{X}_1,\ldots,\mathrm{X}_\mathrm{d})$ be a *d*-dimensional vector, corresponding to the features of the instances.

Let the set of unlabeled examples be:

$$D_2 = \{x_j\}_{j=l+1}^{l+u}.$$

u is the number of unlabeled examples. For these examples, the classification/regression function is unknown; these instances have not been classified by an expert in the field.

2.1 Supervised Learning

Definition 4. *Supervised learning*. Let the domain of instances be χ, and the domain of labels be γ. Given a training set of *l* independently and identically distributed examples $D_1 = \{(x_i, y_i)\}_{i=1}^{l}$, supervised learning trains a function $f : \chi \to \gamma$, $f(x)$ predicts the label *y* on future data *x*.

Depending on the domain of label *y*, supervised learning problems are further divided into classification and regression.

Definition 5. *Classification.* Classification is a supervised learning problem with discrete classes *Y*. The function *f* is called a classifier. Classification identifies the category a new observation belongs to, based on a classifier, which is learned from a training set of instances.

For example, when we want to forecast the day temperature and we are not interested in the exact temperature but in the category it belongs to, we have to learn a classifier with predefined values for the classification function:

$$Y = \{low, medium, high\}$$

Definition 6. *Regression*. Regression is a supervised learning problem with continuous *Y*. The function *f* is called a regression function. Regression analysis estimates the conditional expectation of Y given the values of the attributes $X_1,..., X_d$.

Considering the temperature example above, if we want to know the exact value of the temperature (for instance: 19.7° C, 15.8° C, etc), a regression function has to be learned.

Definition 7. *Root Mean Squared Error* (RMSE)

RMSE measures the differences between the predicted values and the observed ones with respect to the average of all examples in the test set.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} c(x_i, y_i, f(x_i))}{n}} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - f(x_i))^2}{n}} \quad (2.1)$$

Definition 8. *Loss function*. A loss function $c(x, y, f(x)) \in [0, \infty)$ measures the amount of loss of the prediction. It quantifies the amount by which the prediction deviates from the actual values.

Definition 9. *Risk*. The risk associated with *f* is defined as the expectation of the loss function: $R(f) = E[c(x, y, f(x))]$.

Definition 10. *Emperical Risk* - in general, the risk $R(f)$ cannot be computed because the distribution $P(x, y)$ is unknown to the learning algorithm. However, we can compute an approximation to it, called empirical risk. The empirical risk of a function is the average loss of *f* on a labeled training set.

$$R_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} c(x_i, y_i, f(x_i)) \quad (2.2)$$

Definition 11. *Regularization* in the field of machine learning refers to a process of introducing additional information in order to prevent overfitting.

Definition 12. *Regularized Risk*. A regularizer $\Omega(f)$ is a non-negative function, a penalty term, which has a non-negative real-valued output. If *f* is "smooth", $\Omega(f)$ will be close to zero. If *f* is too zigzagged and overfits the training data, $\Omega(f)$ will be large. We can define the regularized risk as the sum of the emperical risk and a regularizer:

$$Err(f) = R_{emp}(f) + \lambda \Omega(f) \quad (2.3)$$

The concept of regularization was first introduced by Tinkhov [58] for model selection. The best model is defined by a set of parameters that optimize an objective function of the form presented in (2.3).

Usually, the following regularizers are used (*w* is the model's weight vector):

- *L2 norm:* $\Omega(f) = \| w \|_2^2 = \left(\sqrt{\sum_{s=1}^{d} w_s^2} \right)^2 = \sum_{s=1}^{d} w_s^2$ (in ridge regression [36]);

- *L1 norm:* $\Omega(f) = \| w \|_1 = \sum_{s=1}^{d} w_s$ (lasso method [37])

2.1.1 Naive Bayes Classifier

The naïve Bayes classifier [66, 92] is a simple probabilistic supervised classifier. It makes use of the Bayes' formula:

$$P(y_j \mid a_1,..,a_m)P(a_1,..,a_m) = P(a_1,..,a_m \mid y_j)P(y_j)$$

$$P(y_j \mid a_1,..,a_m) = \frac{P(a_1,..,a_m \mid y_j)P(y_j)}{P(a_1,..,a_m)} \tag{2.4}$$

In order to classify new examples the naive Bayes classifier chooses the hypothesis that is most probable (*maximum a posteriori*). For each y_i, we calculate $P(y_j \mid a_1,..,a_m)$ and return the most probable classification.

Because the maximum does not depend on $P(a_1,..,a_m)$:

$$\arg\max_{y_j} \frac{P(a_1,..,a_m \mid y_j)P(y_j)}{P(a_1,..,a_m)} = \arg\max_{y_j} P(a_1,..,a_m \mid y_j)P(y_j) \tag{2.5}$$

The corresponding classifier is the function *f** defined as:

$$f^* = \arg\max_{y_j} P(y_j)P(a_1,..,a_m \mid y_j) \tag{2.6}$$

The naïve Bayes classifier relies on the preposition that the attributes are conditionally independent given the class. *Independence*:

$$P(a_1,..,a_m \mid y_j) = \prod_{k=1}^{m} P(a_k \mid y_j) \tag{2.7}$$

$$f^* = \arg\max_{y_j} P(y_j)\prod_{k=1}^{m} P(a_k \mid y_j) \tag{2.8}$$

To calculate $P(a_k \mid y_j)$:

- when the attribute A_k has discrete values, we can compute the proportion of examples for which the value of attribute A_k is a_k.

$$P(a_k \mid y_j) = \frac{\#examples: A_k = a_k \text{ and } Y = y_j}{\#examples: Y = y_j} \tag{2.9}$$

- when the attribute A_k is real-valued, in order to calculate $P(a_k \mid y_j)$, we can assume the attribute A_k is normally distributed.

$$P(a_k \mid y_j) = N(a_k, \mu_y, \sigma_y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e \frac{-(a_k - \mu_y)^2}{2\sigma_y^2} \tag{2.10}$$

The naïve Bayes classifier is fast and it does not require complex computations like reversed matrices and iterative procedures. It is a popular method for text classification [48] and has been applied to spam filtering [18].

2.1.2 Bayes Classifier, based on Multivariate Normal Distribution

The multivariate normal distribution is often used to describe any set of correlated real-valued random variables each of which clusters around a mean value. Let the parameter of the model be $\theta = (\pi, \mu, \Sigma)$:

- π - prior class probability. Originally, for each class *j*, it is the proportion of instances with this classification.
- μ - mean vector.
- ∑ - covariance matrix.

When the examples in the dataset are independent and identically distributed:

$$P(D \mid \theta) = \prod_{i=1}^{l} P(x_i, y_i \mid \theta) = \prod_{i=1}^{l} P(y_i \mid \theta) P(x_i \mid y_i, \theta) \tag{2.11}$$

In order to find the local maximum and the parameter θ, we set the derivative of $P(D \mid \theta)$ to 0 and find $\theta = (\pi, \mu, \Sigma)$:

$$\pi_j = \frac{l_j}{l} \tag{2.12}$$

$$\mu_j = \frac{1}{l_j} \sum_{i: y_i = j} x_i \tag{2.13}$$

$$\Sigma_j = \frac{1}{l_j} \sum_{i:y_i=j} (x_i - \mu_j)(x_i - \mu_j)^T \tag{2.14}$$

l_j is the number of examples having classification y_j and l is the number of all labeled examples.

This is a Bayesian classifier and to classify a new example x_i, the value of $P(y_i \mid x_i, \theta)$ should be calculated for each class y_i. The prediction of x_i is the y_i for which the probability $P(y_i \mid x_i, \theta)$ is maximal (*maximum a posteriori*).

$$P(y_i \mid x_i, \theta) = \frac{P(y_i)P(x_i \mid \theta, y_i)}{P(x_i)} = \frac{P(y_i)N(x_i, \mu_y, \Sigma_y)}{P(x_i)} \tag{2.15}$$

This classifier differs from the naïve Bayes classifier in not assuming that the attributes are conditionally independent given the class, but using a multivariate normal distribution instead:

$$P(x_i \mid \theta, y_i) = N(x_i, \mu_y, \Sigma_y) \tag{2.16}$$

$$N(x, \mu_y, \Sigma_y) = \frac{1}{(2\pi)^{\frac{D}{2}} \mid \Sigma y \mid^{\frac{1}{2}}} e^{\frac{-(x-\mu_y)^T \Sigma_y^{-1} (x-\mu_y)}{2}} \tag{2.17}$$

The multivariate normal distribution is data dependent. When calculating the covariance matrices from a dataset, their ranks may not be full. This is the so-called degenerate case. The simplest solution is to "smooth" the covariance matrix by adding a weighted identity matrix to make it a full rank one. Alternatively, we can perform a dimensionality-reduction procedure (for example Principal Component Analysis – PCA [19]) to make sure each cluster has a full rank covariance matrix.

Both the naïve Bayes classifier and the supervised classifier based on multivariate normal distribution are used and evaluated in the proposed semi-supervised multi-view segmentation learning system.

2.1.3 Support Vector Machines

A support vector machine (SVM) constructs a hyperplane in a high dimensional space, which can be used for classification or regression. A good separation is achieved by the hyperplane that has the largest distance to the nearest

training data point of any class. For example, in Fig. 1 below, 4 is a better separator than 1, 2 or 3 because it will be more likely to classify correctly new examples that are close to the current decision boundary.

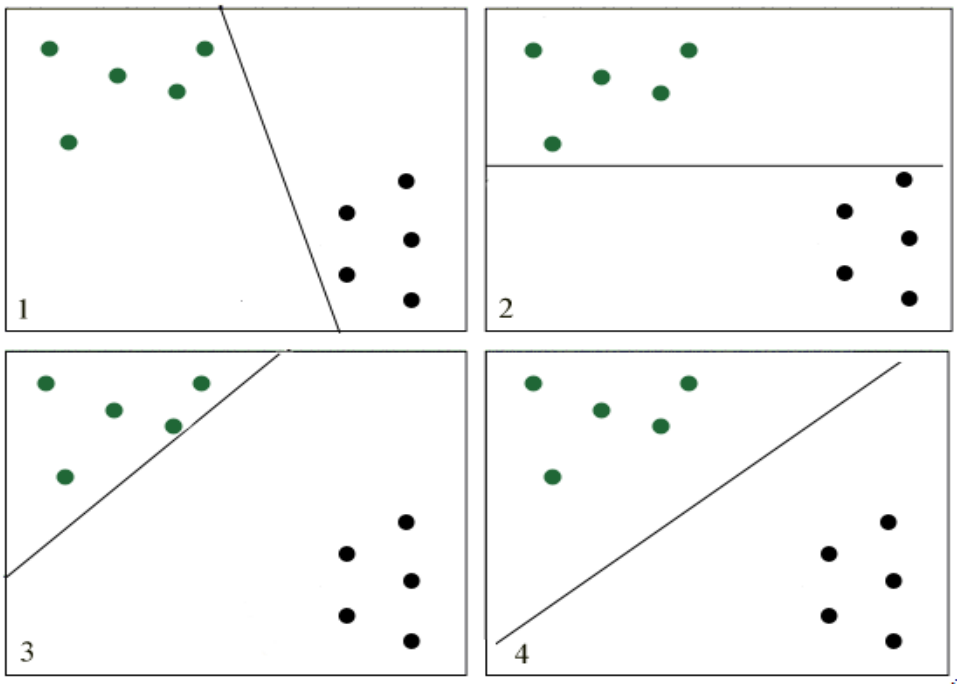


Fig. 1 Decision boundary - 1,2,3 and 4

Let $y \in \{1,-1\}$ and $f(x) = w^T x + b$

Definition 13. *A decision boundary* is defined by the set:

$$\{x \mid w^T x + b = 0\} \tag{2.18}$$

The prediction of the label of x depends on the $sign(f(x))$.

Definition 14. *The absolute value of the distance* from an instance x to the decision boundary is:

$$\frac{|f(x)|}{\|w\|} \tag{2.19}$$

Definition 15. *Signed distance* of a labeled instance to the decision boundary:

$$\frac{yf(x)}{\|w\|} \tag{2.20}$$

Let's assume that the training data is *linearly separable*, there is at least one linear decision boundary that can separate all labeled examples so that they are on the correct side of the boundary. Then, the signed distance is positive when:

- a positive instance is on the positive side
- a negative instance is on the negative side.

We want to find the decision boundary that is as far from the closest element and maximizes:

$$\max_{w,b} \min_{i=1}^{l} \frac{y_i f(x_i)}{\|w\|} \tag{2.21}$$

From all points, the closest one is situated at "minimum" distance to the decision boundary and we need a decision boundary that is as far as possible from the closest point. (2.21) is equivalent to minimizing:

$$\begin{aligned} &\min_{w,b} \|w^2\| \\ &subject\ to: y_i(w^T x_i + b) \geq 1, i = 1,...,l \end{aligned} \tag{2.22}$$

Definition 16. *Hinge loss function:*

$$c(x, y, f(x)) = \max(1 - y_i(w^T x_i + b), 0) \tag{2.23}$$

In order to broader the number of problems we can learn with SVM (we want to learn even linearly non-separable problems), let's relax this assumption and allow $y_i f(x_i) < 1$ for some labeled examples. Then, the problem changes to find the minimum of:

$$\min_{w,b} \sum_{i=1}^{l} \max(1 - y_i(w^T x_i + b), 0) + \lambda \|w\|^2 \tag{2.24}$$

The hinge loss penalizes examples that are on the right side of the boundary, but $0 \leq yf(x) \leq 1$. It further penalizes instances that are on the wrong side of the boundary and $yf(x) < 0$.

2.2 Unsupervised Learning

Clustering [20] is the process of dividing objects into groups. A cluster comprises of elements, which are similar to one another. The elements of one cluster are expected to differ widely from the elements of another cluster.

A human learns to differentiate between multiple categories - dogs, cats, food, etc. at a very young age. Similarly, an autonomous agent performing unsupervised learning should be able to cluster objects, based on their features and similarities.

2.2.1 kMeans

kMeans [20, 66] is a clustering algorithm. It divides the data into *k* predefined clusters. Beforehand, a person should decide on the number of clusters included.

kMeans creates new objects - centroids c_j, which are representative of the clusters.

At first the algorithm chooses randomly *k* existing objects and defines them as centroids - one centroid for each cluster. For each of the remaining objects it calculates the distance to the *k* centroids. An object is assigned to the cluster, whose centroid is closest. After that, the values of the centroids should be refreshed. C_j stands for cluster *j* and c_j is the centroid of C_j.

$$c_j = \frac{\sum_{x_{ij} \in Cj} x_{ij}}{|Cj|} \tag{2.25}$$

The process is an iterative procedure, which finishes when:

$$eps = \sum_{i=1}^{k} \sum_{p \in Ci} d^2(p, c_j) \tag{2.26}$$

or a predefined value - maximum number of iterations is reached. $d(p,c_j)$ is a distance function and measures how far an element *p* is from the centroid c_j.

2.2.2 Expectation-Maximization Algorithm

The Expectation-Maximization algorithm (EM-algorithm) [88] is an iterative procedure, an unsupervised algorithm, which finds local maxima of $\log(D|\theta)$ or

$\log(D|\theta)+\log(\theta)$ using the maximum likelihood method. The global maximum in the probability space is not guaranteed. As a result, the elements in the training set D are clustered in "internally inferred groups" or the so-called clusters.

1. Initialization: t = 0

$$D=\{x_i\}_{i=1}^{l}$$

$$\theta^{(0)}=\{\pi_j^{(0)},\mu_j^{(0)},\Sigma_j^{(0)}\}$$

$$N(x,\mu_y,\Sigma_y)=\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma y|^{\frac{1}{2}}}e\frac{-(x-\mu_y)^T\Sigma_y^{-1}(x-\mu_y)}{2} \quad (2.27)$$

$$\pi_0=\frac{l_j}{l} \quad (2.28)$$

$$\mu_0=\frac{1}{l_j}\sum_{i:yi=j}x_i \quad (2.29)$$

l_j is the initial number of labeled examples, whose label is j. The EM-algorithm performs two steps: an **Expectation** step - expectation of cluster probabilities (which are the "expected" class values) and a **Maximization** step - maximization of the likelihood of the distributions, given the data.

2. While $P(D|\theta^t)$ does not change significantly

- Expectation step (E-step): given the current estimate of the parameters θ^t, find the conditional distribution of Y_i. Probabilistically re-label the examples based on θ^t;

$$\gamma_{ij}=P(y_j|x_i,\theta^{(t)})=\frac{\pi_j^{(t)}N(x_i,\mu_j^{(t)},\Sigma_j^{(t)})}{\sum_k\pi_k^{(t)}N(x_i,\mu_k^{(t)},\Sigma_k^{(t)})} \quad (2.30)$$

- Maximization step (M-step): re-estimate the new parameter θ^{t+1}.

$$\pi_j^{(t+1)}=\frac{\sum_{i=1}^{l}\gamma_{ij}}{l} \quad (2.31)$$

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{l} \gamma_{ij} x_i}{\sum_{i=1}^{l} \gamma_{ij}} \tag{2.32}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{l} \gamma_{ij} (x_i - \mu_j^{(t+1)})(x_i - \mu_j^{(t+1)})^T}{\sum_{i=1}^{l} \gamma_{ij}} \tag{2.33}$$

2.3 Semi-supervised Learning

Recently, there has been significant interest in semi-supervised learning [2, 15, 78]. It requires less human effort and makes use of unlabeled instances. Semi-supervised learning falls between unsupervised learning and supervised learning. An expert in the field has already labeled a small amount of instances (D_1). Unlabeled instances (D_2) are also used and added to the pool of training examples. The final training data contains the examples of D_1 and D_2 ($D = D_1 \cup D_2$)

Let the number of labeled examples be l and the number of the unlabeled examples - u. $D = D_1 \cup D_2$, $D_1 = \{(x_i, y_i)\}_{i=1}^{l}$, $D_2 = \{x_j\}_{j=l+1}^{l+u}$.

There are two types of semi-supervised learning: *inductive* and *trunsductive* semi-supervised learning.

Definition 8. *Inductive semi-supervised learning*. Given a training set of examples D, inductive semi-supervised learning learns a function $f : \chi \to \gamma$ so that f is expected to be a good prediction algorithm for future data, beyond $D_2 = \{x_j\}_{j=l+1}^{l+u}$.

Definition 9. *Transductive semi-supervised learning*. Given a training set of examples D, transductive semi-supervised learning trains a function $f : \chi^{l+u} \to \gamma^{l+u}$ so that f is expected to be a good prediction algorithm only for the unlabeled examples $D_2 = \{x_j\}_{j=l+1}^{l+u}$.

f does not make predictions outside the set of unlabeled examples.

2.3.1 Self-Training

Self-training [14, 89] refers to a classifier whose learning phase uses its own predictions to teach itself. At first, only the labeled instances are used for the learning of the classifier (supervised learning, based on D_1). After that, this classifier predicts the labels of the unclassified instances (D_2). A portion of the newly labeled examples (former unlabeled) augments the set of labeled examples and the classifier is retrained. Once again, new predictions are assigned to the remaining unlabeled instances. It is an iterative procedure, which finishes either when some maximum number of iterations is reached, or some other convergent condition becomes true.

A modification of the algorithm at each step augments the set of labeled examples only with the most credible ones, predicted to be the most probable ones by the classifier, not only a random portion of the newly-classified examples.

```
void Self Training()
{
        labeled = D1;  // Initialization
        while (stop criterion is false)
        {
                //  supervised learning based on "labeled"
                L1 = learnSupervised(labeled);
                // label all unlabeled examples and return a subset
                of the ones the classifier is most confident about.
                mostConfident = L1(D2);
                D2 -=  mostConfident;
                labeled += mostConfident;
        }
}
```

Self-training Assumption

The assumption of self-training is that its own predictions, at least the most reliable ones, tend to be correct. If the original classifier is too weak (L_1), we cannot expect considerable improvement in the accuracy. Therefore, the originally learned classifier should be a decent one.

2.3.2 Cluster-then-label

"*Cluster-then-label*" [15] is a semi-supervised algorithm, which uses labeled and unlabeled examples, applies both unsupervised and supervised techniques in order to find the classifications of the unlabeled examples. It first clusters the instances (labeled and unlabeled) into *k* groups, performing unsupervised clustering algorithm. After that, for each cluster Cj - based on the labeled examples in it, a supervised algorithm is learned and used to classify the unlabeled examples, which belong to Cj. If there are no labeled examples in a cluster, the algorithm uses all labeled instances to learn the classification/regression function.

```
void clusterThenLabel()
{
        // cluster all examples (in K clusters)
        clusterK(D);
        Sj = labeledInCluster(j);
        for (int j = 0; j < K; j++)
        {
                // If Sj contains no labeled examples
                if(containsNoLabeled(Sj))
                {
                        // use all labeled data.
                        fj = learn(D1);
                }
                else
                {
                        // use the labeled examples in Sj to learn fj.
                        fj = learn(Sj);
```

```
            }
            //classify the unlabeled examples in Sj, based on fj
            predict(Sj, fj);
        }
    }
```

2.3.3 Semi-supervised Expectation-Maximization algorithm

The semi-supervised EM-algorithm is a modification of the EM-algorithm. It is an iterative procedure and the likelihood is maximized using labeled and unlabeled examples [2].

1. Initialization: t = 0. Based on the labeled examples, we find $\theta^{(0)}$

$$\theta^{(0)} = \{\pi_j^{(0)}, \mu_j^{(0)}, \Sigma_j^{(0)}\}$$

$$D_1 = \{(x_i, y_i)\}_{i=1}^{l}$$

$$D_2 = \{x_j\}_{j=l+1}^{l+u},$$

$$\pi_0 = \frac{l_j}{l} \tag{2.34}$$

$$\mu_0 = \frac{1}{l_j} \sum_{i:yi=j} x_i \tag{2.35}$$

$$\Sigma_0 = \frac{1}{l_j} \sum_{i:yi=j} (x_i - \mu_j)(x_i - \mu_j)^T \tag{2.36}$$

2. While $P(D \mid \theta^t)$ does not change significantly – compared to $P(D \mid \theta^{t-1})$ in the previous step.

- E-STEP - find the hidden distribution $P(D_2 \mid D_1, \theta^t)$
  - for the unlabeled instances in D_2 (as in the unsupervised EM):

$$\gamma_{ij} = P(y_j \mid x_i, \theta^{(t)}) = \frac{\pi_j^{(t)} N(x_i, \mu_j^{(t)}, \Sigma_j^{(t)})}{\sum_k \pi_k^{(t)} N(x_i, \mu_k^{(t)}, \Sigma_k^{(t)})} \tag{2.37}$$

  - for the labeled instances D_1:

$$\gamma_{ij} = \begin{cases} 1 \ \textit{if} \ y_i = j \\ 0 \ \textit{otherwise} \end{cases} \tag{2.38}$$

- M-STEP – find θ^{t+1}, using γ_{ij}

$$\pi_j^{(t+1)} = \frac{l_j}{l+u} \tag{2.39}$$

$$l_j = \sum_{i=1}^{l+u} \gamma_{ij} \tag{2.40}$$

$$\mu_j^{(t+1)} = \frac{1}{l_j} \sum_{i=1}^{l+u} \gamma_{ij} x_i \tag{2.41}$$

$$\Sigma_j^{(t+1)} = \frac{1}{l_j} \sum_{i=1}^{l+u} \gamma_{ij} (x_i - \mu_j^{(t+1)})(x_i - \mu_j^{(t+1)})^T \tag{2.42}$$

Kamal Nigam, Andrew McCallum and Tom Mitchell [16] perform semi-supervised text classification using the EM-algorithm. They further demonstrate that deterministic annealing [16], a variant of EM, can overcome the problem of local maxima and increase the classification accuracy.

2.3.4 Semi-supervised Support Vector Machines

A semi-supervised support vector machine - SSSVM [85, 86, 100] uses both labeled and unlabeled examples. Since labeled examples do not have labels, we do not know on which side of the boundary they are.

Let's define the function y` as:

$$y` = sign(f(x)) \tag{2.43}$$

Definition 15. *Hat loss function:*

$$c(x, y`, f(x)) = \max(1 - y`(w^T x_i + b), 0) = \max(1 - |w^T x_i + b|, 0)) \tag{2.44}$$

Thus, an unlabeled instance x is always on the correct side of the decision boundary. It prefers examples for which $f(x) \geq 1$ *or* $f(x) \leq -1$. These are the examples that are far from the boundary. The hat loss penalizes examples for which: $-1 < f(x) < 1$.

We now incorporate the hat loss on the unlabeled data $D_2 = \{x_j\}_{j=l+1}^{l+u}$.

Regularizer:

$$\Omega(f) = \lambda_1 \|w\|^2 + \lambda_2 \sum_{j=l+1}^{l+u} \max(1 - |w^T x_j + b|, 0) \tag{2.45}$$

Finally, we have to find the following minimum:

$$\min_{w,b} \sum_{i=1}^{l} \max(1 - y_i(w^T x_i + b), 0) + \lambda_1 \|w\|^2 + \lambda_2 \sum_{j=l+1}^{l+u} \max(1 - |w^T x_j + b|, 0)$$

$$subject\ \ to \frac{1}{u} \sum_{j=l+1}^{l+u} w^T x_j + b = \frac{1}{l} \sum_{i=1}^{l} y_i \tag{2.46}$$

When the optimization is over, the algorithm has learned the parameter *b* and the vector *w*. It corresponds to the decision boundary. Based on this boundary we predict the labels of future examples.

2.3.5 Graph-based Semi-supervised Learning

Graph-based semi-supervised learning [44, 79, 84] constructs a graph from the training examples. The nodes of the graph are data points (labeled and unlabeled) and the edges represent similarities between points.

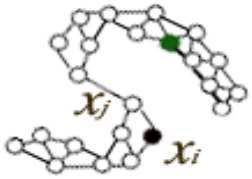
Graph elements:

- nodes (vertices) – all examples in the dataset:

$$D_1 = \{(x_i, y_i)\}_{i=1}^{l}$$
$$D_2 = \{x_j\}_{j=l+1}^{l+u}$$

- edges – an edge represents the similarity between two vertices, a missing edge corresponds to infinite distance.

Known labels are used to propagate information through the graph in order to label all nodes. In Fig. 2 the green and the black vertices are labeled examples (two classes). We should assign *y* values to the white vertices (unlabeled examples). Edges are constructed, connecting neighboring vertices.



- Labeled examples: green and black color
- Unlabeled examples: white color

Fig. 2 A semi-supervised graph

An edge between two vertices represents the *similarity* (w_{ij}) between them. The closer two vertices are, the higher the value of w_{ij} is.

Euclidean distance:

$$\|x_i - x_j\| = \sqrt{\sum_s (x_{is} - x_{js})^2} \tag{2.47}$$

The Gaussian function is used for the weights. It is also called a Gaussian kernel or a Radial Basis Function (RBF) kernel.

$$w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}} \tag{2.48}$$

$$w_{ij} = \begin{cases} 1 & when \quad x_i = x_j \\ 0 & when \ \|x_i - x_j\| \ approaches \infty \end{cases} \tag{2.49}$$

Mincut algorithm

Mincut [38, 96] is a graph cut problem. The objective is to find a minimum set of edges whose removal blocks the whole flow from one of the classes to the other class. This defines a "cut," or a partition of the graph into two sets of vertices, each corresponding to one of the classes.

Let $f(x) \in \{1,-1\}$ and $f(x_i) = y_i$ for the labeled instances.

The cut size should be minimized:

$$\sum_{i,j: f(x_i) \neq f(x_j)} w_{ij} \tag{2.50}$$

If an edge w_{ij} is removed, the two vertices are too far away and the distance between them is high. As we have only two possible class values for $f(x)$, we remove an edge if $f(x_i) \neq f(x_j)$.

Loss function:

$$c(x, y, f(x)) = \infty(y - f(x))^2 \tag{2.51}$$

$$c(x, y, f(x)) = \begin{cases} 0 \quad when \quad y = f(x) \\ \infty otherwise \end{cases} \tag{2.52}$$

Regularizer:

$$\Omega(f) = \sum_{i,j=1}^{l+u} \frac{w_{ij}(f(x_i) - f(x_j))^2}{4} \tag{2.53}$$

- If x_i and x_j are not connected, then $w_{ij} = 0$
- If the edge exists and is not cut, then $f(x_i) - f(x_j) = 0$

So, we have to find the following minimum:

$$\min_{f: f(x) \in \{1,-1\}} \sum_{i=1}^{l} \infty(y_i - f(x_i))^2 + \sum_{i,j=1}^{l+u} w_{ij}(f(x_i) - f(x_j))^2 \tag{2.54}$$

Mincut is a transductive learning algorithm. It learns a function f that is restricted to the labeled and unlabeled vertices in the graph. If an example is not in the training set, we cannot predict its label, unless we insert it as a new vertex and repeat the procedure.

Harmonic function

Definition 16. *A harmonic function* is a function that can be defined as:

$$f(x_i) = y_i, i = 1,\ldots,l$$
$$f(x_j) = \frac{\sum_{k=1}^{l+u} w_{jk} f(x_k)}{\sum_{k=1}^{l+u} w_{jk}}, j = l+1,\ldots,l+u \tag{2.55}$$

For the labeled examples the value of the harmonic function is the classification of the instances. For the unlabeled examples it is the estimated

weighted average of the other points and depends on the weights w_{jk}. Therefore, the value of each unlabeled vertex is the weighted average of its neighbors' values.

Let's relax *f* to have real values. We have to find the following minimum:

$$\min_{f:f(x)\in R} \sum_{i=1}^{l} \infty (y_i - f(x_i))^2 + \sum_{i,j=1}^{l+u} w_{ij} (f(x_i) - f(x_j))^2 \tag{2.56}$$

which is equivalent to:

$$\min_{f:f(x)\in R} \sum_{i,j=1}^{l+u} w_{ij} (f(x_i) - f(x_j))^2$$
$$subject\ to: f(x_i) = y_i, i = 1,...,l \tag{2.57}$$

Computing the harmonic function

1. *Initialization*

   For the labeled vertices $i = 1,...,l$ set: $f(x_i) = y_i$

   For the unlabeled vertices, set some randomly generated values.

2. *Do*

   Update each unlabeled vertex's *f* value based on its neighbors:

$$f(x_j) = \frac{\sum_{k=1}^{l+u} w_{jk} f(x_k)}{\sum_{k=1}^{l+u} w_{jk}}, j = l+1,...,l+u$$

This iterative procedure is guaranteed to converge to the harmonic function, regardless of the initial values of the unlabeled vertices. It is also called *label propagation*, it "propagates" labels from the labeled vertices through the edges to all the unlabeled vertices.

2.4 Semi-supervised Multi-view Learning

2.4.1 Multiple Sources of Information

In our world, a single object can have multiple representations, i.e. projections on different sources of information. We will call these sources - *views*. Generating different views corresponds to partitioning a set of attributes into

multiple views. It decomposes the original set of features into multiple disjoint subsets. Different examples of data sources can be seen in Fig. 3. For instance, a movie can have the following two sources of information: a video and an audio representation.

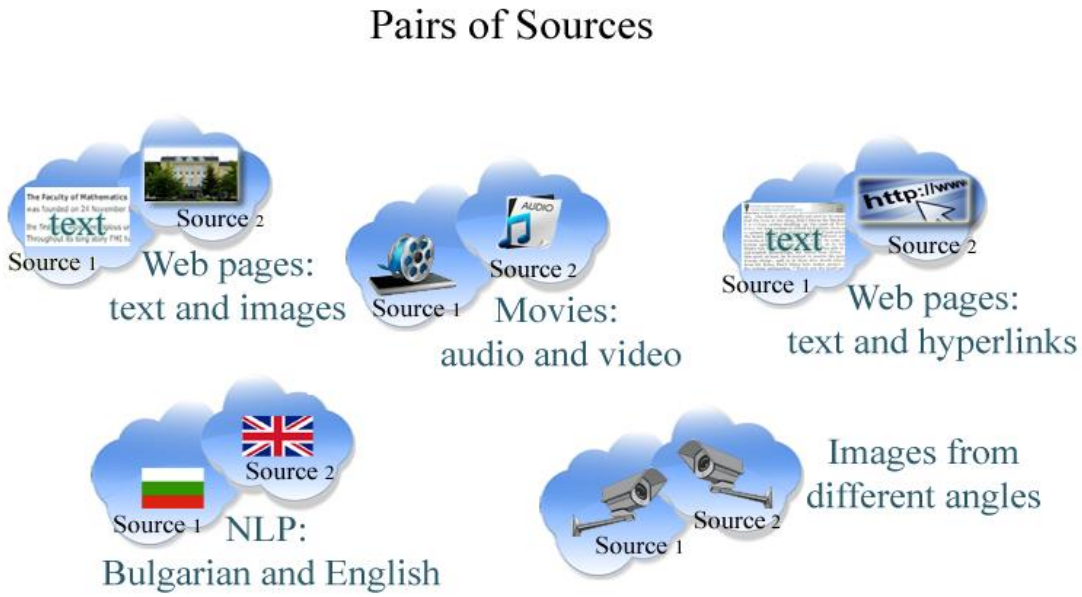


Fig. 3 Examples of multiple sources of data

Since 1998, when Blum and Mitchell [3] published the core co-training algorithm, there has been a booming development of algorithms and applications in this field. Blum and Mitchell classified web pages (WebKB course dataset) in which the description of each example can be partitioned into two distinct views:

- the words occurring on the web page
- the words occurring in hyperlinks which point to that page

Kamal Nigam and Rayid Ghani [4, 98] also explore the WebKB dataset and analyze the effectiveness and applicability of the co-training algorithm. The same dataset has been further explored by Vikas Sindhwani, Partha Niyogi and Mikhail Belkin [5, 72], who propose a co-regularization approach to semi-supervised learning.

Languages like Chinese suffer from insufficiency of labeled examples. Xiaojun Wan [6, 74] uses a co-training bilingual approach for improving the prediction on Chinese movie reviews. Each review has:

- an English view
- a Chinese view

Besides these applications in the field of natural language processing, co-training has also been applied in computer vision. To identify cars Anat Levin, Paul Viola, and Yoav Freund [7] use the following two views:

- grey-level of an image
- background-subtracted images

Co-training has also been applied for "co-tracking" [8] – with the following two views:

- histogram of oriented gradients (HOG);
- color histogram.

Cheng and Wang [9] propose a new SVM algorithm – "Co-SVM", which uses a co-training approach and achieves better results than a normal SVM on classifying images using color and texture as separate views.

Sonal Gupta, Joohyun Kim, Kristen Grauman and Raymond Mooney [10] show how to learn about human activities based on both:

- visual content (videos and images);
- spoken commentary (associated text).

C. Christoudias, R. Urtasun, and T. Darrell [54, 55] defined a view disagreement problem. Noise in the views can influence the performance of the multi-view learning algorithms. The examples of each view do not always belong to the same class. They sometimes belong to an additional background class because of noise. In their work they identify a speaker in a video, using the following views:

- an audio view;
- a video view.

A simple way to convert from a single view to multiple views is to split the original attributes into different views at random. Following this idea, Brefeld and Scheffer [11] present a co-EM version of the Support Vector Machine without an explicit multi-view representation. However, dividing the feature set into multiple views in a way that adheres to multi-view learning is not a simple task, and is dependent on both the learner and the data domain.

2.4.2 Principles of Multi-view Learning

Why should two classifiers be learnt and not just one, based on the concatenate features of the different views? When is multi-view learning effective? On a small training set, the concatenation of features causes overfitting. Semi-supervised multi-view learning penalizes functions which overfit the training data. In statistics, this process is also known as regularization and is especially applicable when we have a small amount of labeled instances. Additional regularization terms, which rely on multiple sources of information, are added to the error of the learner.

In order semi-supervised multi-view learning to be effective, the following principles [13] should hold:

- *Consensus Principle*

Multi-view learning depends on multiple hypotheses, each of which is learned based on a different source of information, and the agreement among them plays a major role during the learning phase. By minimizing the disagreement rate of the multiple models, the error rate of each hypothesis is minimized [12].

- *Complementary Principle*

The complementary principle states that each source of information should contain some knowledge that other sources do not offer. This complementary information, provided by the underlying multiple views, can be exploited to improve the learning performance.

2.4.3 Problem Formulation

In semi-supervised learning there are :

- labeled examples: $D_1 = \{(x_i, y_i)\}_{i=1}^{l}$
- unlabeled examples: $D_2 = \{x_j\}_{j=l+1}^{l+u}$

$$D = D_1 \cup D_2$$

In semi-supervised multi-view learning each example $x = (x^{(1)},...,x^{(k)})$ has multiple views: $x^{(1)} \in X^{(1)},...,x^{(k)} \in X^{(k)}$. Each view consists of characteristics of the object, projected onto its data source. We learn a separate classifier on each view and the process of prediction is a combination of multiple learners.

Definition 17: *Semi-supervised multi-view classification:* the goal is to learn a tuple $f = (f_1^{(1)},..., f_k^{(k)})$ based on *D*:

$$f^{(1)} : X^{(1)} \to \{1,...,k\}$$
$$...$$
$$f^{(k)} : X^{(k)} \to \{1,...,k\}$$

Definition 18: *Semi-supervised multi-view regression:* the goal is to learn a tuple $f = (f_1^{(1)},..., f_k^{(k)})$ based on *D*:

$$f^{(1)} : X^{(1)} \to R$$
$$...$$
$$f^{(k)} : X^{(k)} \to R$$

In order to classify new instances, we have to combine $f_1^{(1)},..., f_k^{(k)}$ so that a final prediction is made.

It is important to note that unlike other ensemble methods [70], semi-supervised multi-view learning [71, 73, 76, 87] does not learn $f_1^{(1)},..., f_k^{(k)}$ independently. Usually, they teach each other in the process of learning or a common loss is optimized so that the final prediction based on all views is minimized (Fig. 4).

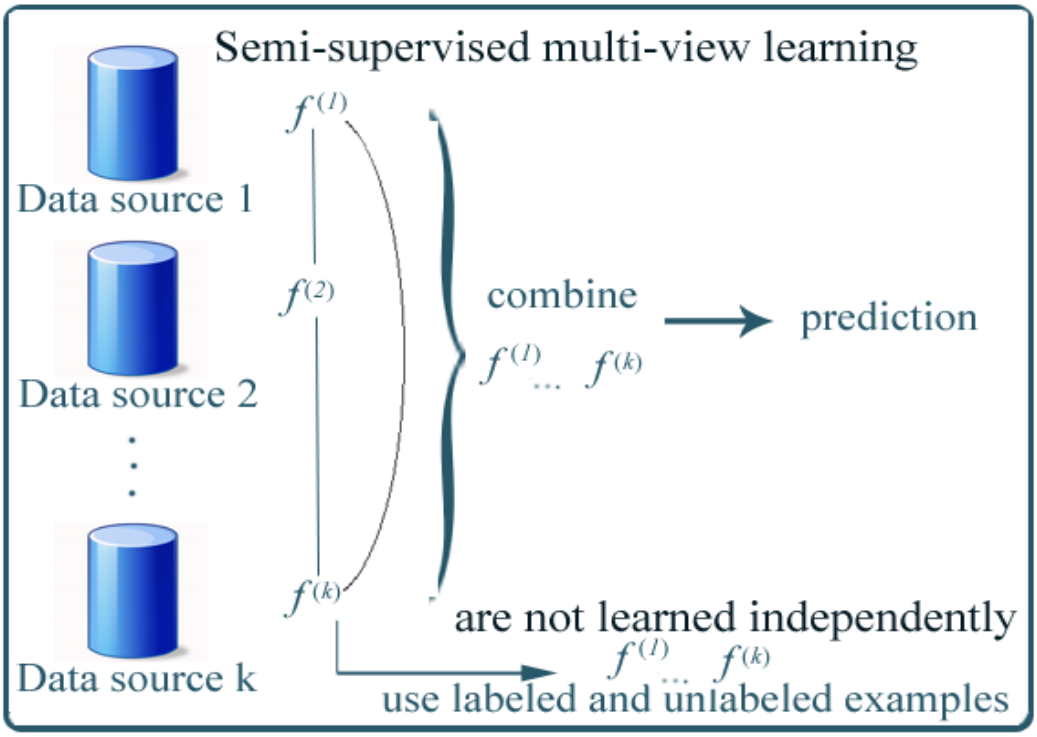


Fig. 4 Semi-supervised multi-view learning

2.4.4 Co-training

The original co-training algorithm [3, 75, 77, 97, 99] was proposed by Blum and Mitchell. They used the algorithm for faculty web-page classification. The first view contains the words on the web-pages and the second – the links that point to the web pages. Out of the 1051 web pages only 12 labeled examples were used and an error rate of 5% was achieved.

Since 1998, when the algorithm was published, there has been considerable interest in the whole field of semi-supervised multi-view learning. This algorithm inspired the scientific society and has been cited more than 3650 times. It proved to be useful and has been applied in many areas – not only for web page classification, but also for "co-tracking" [8] for identifying cars [7], e-mail classification [69] and statistical parsing [46].

The scientific contributions referring to this algorithm are numerous. Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, Bharat Rao [40] developed a Bayesian undirected graphical model for co-training and a novel co-

training kernel for Gaussian process classifiers. Bickel and Scheffer [41] further developed the co-training algorithm for data clustering. Wang and Zhou [42] studied why the co-training algorithm can succeed when there are no redundant views.

Co-training uses two classifiers (L1 and L2). Let U1 and U2 correspond to $view_1$ and $view_2$ of the examples in the training set but only those that have labels. The algorithm augments the set of labeled examples of each classifier, based on the other learner's predictions. Both classifiers are expected to agree on new example labels.

Co-training has proven to be successful in many areas where a natural split in more than one view exists. It outperforms a single standard learner when the following criteria are met [3]:

Co-training Assumptions

(1) Each view (set of features) is sufficient for classification;

(2) The two views (feature sets of each instance) are conditionally independent given the class (2.58).

$$P(X_1 | Y, X_2) = P(X_1 | Y)$$
$$P(X_2 | Y, X_2) = P(X_2 | Y) \quad (2.58)$$

Practically, it is very uncommon to see such uncorrelated views. Even the words in the web-page example above do not fully follow the rule. It is not often possible to find a natural split, division of the attributes, and it depends on the data. Even when this condition does not fully hold, co-training proves to be effective.

Original Algorithm:

1. Learn L1 using U1

   Learn L2 using U2

   Probabilistically label all unlabeled examples using L1. Add L1's most confident examples to U2

   Probabilistically label all unlabeled examples using L2. Add L2's most confident examples to U1

2. Go to 1 until there are no more unlabeled examples or some other stop criterion is met – maximum number of iteration, etc.

The co-training algorithm is a multi-view version of the self-training algorithm, which uses its most credible predictions on unlabeled instances to teach itself. The difference lies in the fact that *two* classifiers are used. These classifiers operate on different views of an example and they teach each other.

Combining the views

Blum and Mitchell proposed a simple framework where the final class probability of a new example is the multiplication of the two views' probabilities (2.59):

$$P(y_j \mid x_i) = P_1(y_j \mid x_i^{(1)}) P_2(y_j \mid x_i^{(2)}) \tag{2.59}$$

2.4.5 Co-EM Algorithm

This algorithm was originally proposed by Nigam & Ghani [4]. It is a hybrid algorithm and an iterative procedure. Like co-training, co-EM trains two classifiers (one EM-algorithm for each view). Like the EM-algorithm, it assigns probabilistic values to unobserved class labels and updates the model parameters.

Let f_1 and f_2 correspond to the classifiers we want to learn in *view1* and *view2* respectively. Initially f_1 is learned based on a Naive Bayes classifier from the labeled examples (supervised learning). Then, f_1 probabilistically labels all the unlabeled data. We train f_2 using the labeled data and the unlabeled data with f_1's labels. f_2 then re-labels the instances so that f_1 can use them again. Co-EM converges as quickly as EM does.

2.4.6 Multi-View Learning, based on error minimization

Each view consists of characteristics of the object, projected onto its data source. Let each instance x consist of multiple views:

$$x = (x^{(1)}, \ldots, x^{(k)}).$$

Let $f_1,..,f_k$ be the regression functions, corresponding to the views $x^{(1)},...,x^{(k)}$. The goal is to find such a combination of learners $f_1^*,..,f_k^*$ so that they can agree with one another and the final combined loss function is minimal. Multiple hypotheses are trained from the same labeled data set, and they are required to make similar predictions on the unlabeled instances.

Multi-view Learning Assumption:

Multiple hypotheses $f_1,..,f_k$ should agree with one another. Furthermore, the agreeing hypotheses should have small regularized empirical risk.

Regularization

In semi-supervised learning we can define $\Omega(f)$ as the sum of the supervised regularizer and the disagreement between the learners on the unlabeled instances, multiplied by λ_2.

$$\Omega(f) = \Omega_{SL}(f) + \lambda_2 \Omega_{SSL}(f) \tag{2.60}$$

$$\Omega_{SSL}(f) = \sum_{u,v=1}^{k} \sum_{i=l+1}^{l+u} c(x_i, f_u(x_i), f_v(x_i)) \tag{2.61}$$

Minimizing the regularized risk.

In the semi-supervised multi-view learning framework, where we have *k* regression functions, we can define *Err* ($f_1,...f_k$) as the sum of the regularized risk of each f_j on $view_j$, based on the labeled examples of D_1, plus the disagreement between the pairs f_u, f_v , based on the unlabeled examples of D_2.

The common error of the *k* learners is defined as:

$$Err(f_1,..,f_k) = \sum_{v=1}^{k} \left(\frac{1}{l} \sum_{i=1}^{l} c(x_i, y_i, f_v(x_i)) + \lambda \Omega_{SL}(f_v)\right) + \lambda_2 \sum_{u,v=1}^{k} \sum_{i=l+1}^{l+u} c(x_i, f_u(x_i), f_v(x_i)) \tag{2.62}$$

Let $f_1^*,..,f_k^*$ be the learners for which *Err* ($f_1,...f_k$) is minimal:

$$f_1^*,..,f_k^* = \arg\min_{f_1,...,f_k} Err(f_1,..,f_k) \tag{2.63}$$

Prediction

The final prediction of a new example is done based on the errors of the *k* views. For classification problems:

$$f(x) = \arg\min_{y \in Y} \sum_{i=1}^{k} c(x, y, f_i^*(x)) \tag{2.64}$$

For each potential class we sum the errors of the multiple learners – f_1^*,.., f_k^* and return the classification for which the sum is minimal.

For regression problems we can sum the predictions of all classifiers and divide by their number - k.

$$f(x) = \frac{1}{k} \sum_{i=1}^{k} f_i^*(x) \tag{2.65}$$

2.4.7 Graph-based Semi-supervised Multi-View Learning

Haibo Li, Yutaka Matsuo, and Mitsuru Ishizuka [59, 60] propose a multi-view semi-supervised learning algorithm, the Co-Label propagation algorithm, which combines information from two sources of information. In the propagation process [61], the label scores of classes are spread between the two views. The proposed algorithm is evaluated using semantic relation classification tasks (recognizing a particular relationship between two or more entities in documents).

The Co-Label propagation algorithm constructs three graphs: graphs G_A and G_B represent the similarities among data points in each view respectively, and the interview graph G_{AB} - a bipartite graph which describes the correlation between data points of two different views.

The intra-view similarity matrices T_A, T_B are $n_x n$ and $s_x s$ similarity matrices constructed from views A and B. W^{AB} and W^{BA} are an $n_x s$ matrix and an $s_x n$ matrix, defined as the correlation matrices between the two views. The inter-view correlation matrices T_{AB}, T_{BA} (normalized) are defined as:

$$T_{ij}^{AB} = \frac{W_{ij}^{AB}}{\sum_{k=0}^{s} W_{ik}^{AB}}$$

$$T_{ij}^{BA} = \frac{W_{ji}^{BA}}{\sum_{k=0}^{n} W_{ki}^{BA}}$$

The Initial label matrices of each view are Y^A_0, Y^B_0

$$(Y_0^A)_{ij} = \begin{cases} 1 \;\; if \;\; f(x_i^{(A)}) = y_j \\ 0 \;\; otherwise \end{cases}$$

$$(Y_0^B)_{ij} = \begin{cases} 1 \;\; if \;\; f(x_i^{(B)}) = y_j \\ 0 \;\; otherwise \end{cases}$$

1. *Propagate each view* - at first, every node in each view receives a contribution from the linked nodes in the same view.

$$Y_{t+1}^{A\,`} = T^A Y_t^A$$

$$Y_{t+1}^{B\,`} = T^B Y_t^B$$

Both labeled and unlabeled nodes are used to create a fully connected intra-view graph in each view. The edge between node *i*, *j* is weighted with a Gaussian kernel.

$$T_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}} \tag{2.66}$$

2. *Propagate between different views* - the second step is to spread label scores among different views.

$$Y_{t+1}^{A\,``} = T^{AB} Y_t^B \tag{2.67}$$

$$Y_{t+1}^{B\,``} = T^{BA} Y_t^A \tag{2.68}$$

3. Combine the label score

$$Y_{t+1}^A = Y_{t+1}^{A\,`} + Y_{t+1}^{A\,``} \tag{2.69}$$

$$Y_{t+1}^{B} = Y_{t+1}^{B}` + Y_{t+1}^{B}`` \tag{2.70}$$

Jingrui He, Rick Lawrence [62] introduce multi-task multi-view learning with both feature heterogeneity and task heterogeneity. They propose a graph-based framework. Within each task, they construct a bipartite graph for each view, modeling the relationship between the examples and the features in the view. The consistency among different views is obtained by requiring them to produce the same classification function.

2.4.8 Multi-View Semi-supervised Support Vector Machines

SVM-2K is a combination of multi-view learning and SVMs which combines the maximum margin and multi-view regularization principles in order to improve the classification performance [63].

Following the manifold regularization [90] approach (exploiting unlabeled examples), Laplacian Support Vector Machines (LapSVMs) have been proposed for semi-supervised classification [64]. Shiliang Sun [65] explores the two-view scenario and uses Laplacian SVMs for image-text classification.

2.4.9 Co-regularized Least Squares

Co-regularized least squares [35] (CoRLS) algorithm is a semi-supervised version of regularized least squares (RLS). The algorithm was first discussed by Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin [39, 82].

Co-regularized least squares use the multi-view approach. The CoRLS algorithm based on labeled and unlabeled examples solves the following minimization problem:

$$(f_1^*, f_2^*) = \arg\min_{f_1, f_2} \frac{1}{l}\sum_{i=1}^{l}(y_i - f_1(x_i^{(1)}))^2 + \frac{1}{l}\sum_{i=1}^{l}(y_i - f_2(x_i^{(2)}))^2 + \lambda\|f_1\|^2 + \lambda\|f_2\|^2 + \lambda_2\sum_{i=l+1}^{l+u}(f_1(x_i^{(1)}) - f_2(x_i^{(2)}))^2 \tag{2.71}$$

According to the representer theorem [43], the minimizer is a linear combination of kernel functions centered on the data. Therefore, the pair of functions have the form:

$$(f_1^*(x^{(1)}), f_2^*(x^{(2)})) = (\sum_{i=1}^{l+u} \alpha_i K^{(1)}(x^{(1)}, x_i^{(1)}), \sum_{i=1}^{l+u} \beta_i K^{(2)}(x^{(2)}, x_i^{(2)})) \quad (2.72)$$

The $(l + u)$-dimensional coefficients α and β can be computed by solving a linear system.

2.5 Genetic Algorithms

Darwin's theory of natural selection [24] revolutionized nineteenth century natural science by revealing that all plants and animals had slowly evolved from earlier live forms [25]. Biology has been the impetus for the development of a highly efficient method for computer optimization – genetic algorithms. Since 1970, when John Holland introduced the concept of genetic algorithms, the discipline has evolved to become one of the most prominent areas in artificial intelligence. The aim is to program such an intelligent program that mimics the biological processes, which occur in nature. The term intelligence refers to the ability to adapt to the changing world – the organism improves and develops new skills, which are necessary for survival.

Genetic algorithms [49, 51] are based on natural evolution and use heuristic search techniques to find decent solutions. In a genetic algorithm, a population of candidate solutions (individuals) is evolved towards better solutions, successively improving a generation of candidate solutions to a given problem, using as a criterion how fit or adept they are at solving the problem. Each candidate solution has a set of features, which can be mutated and altered. The population of individuals is evolving, new individuals arise after crossover of the best fitted so far. New children are born and they replace the worst members of the population [26].

2.5.1 Individuals

All living organisms (individuals) consist of cells, or basic structural units. Each cell contains the same set of one or more chromosomes—strings of DNA. These chromosomes are the features which define the characteristics of the object. In genetic algorithms, chromosomes are often encoded as bit strings - either single bits or short blocks of adjacent bits that represent a particular element of the candidate solution.

2.5.2 Fitness Function

The fitness of an organism is defined as the likelihood for the organism to live and reproduce or as a function of the number of offspring the organism has (fertility). A fitness function measures how close a given design solution is to achieving set aims. A better fitted individual has a higher fitness function and will contribute the most offspring to the next generation.

Function optimization is a common application of genetic algorithms.

$$\max_{a_1,..,a_n} f(a_1,...,a_n)$$

We can use the parameters $a_1, .., a_n$ to form an individual of the form:

a_1	a_2	...	a_n

The goal is to find such a set of values that would maximize or minimize the multi-parameter function. Therefore, in the example above, the fitter the individual, the higher the value of $f(a_1,...,a_n)$.

2.5.3 Selection

This operator selects chromosomes out of the population for reproduction. The fitter the chromosome (based on the fitness function), the more times it is likely to be selected to reproduce. Therefore, individuals with high fitness function will survive, evolve and produce offspring.

2.5.4 Crossover

Crossover of two individuals is the process of reproduction. We strive after population perfection, individuals which are as fitted to their world as possible. Consequently, spreading genes with high fitness is important. Only individuals chosen after the selection process are used for crossover.

There are various existing types of crossover:

- *One-point crossover* (Fig. 5): a random position is selected. The first child inherits the first part of parent 1 and the second part of parent 2. Consequently, the second offspring gets the second part of parent 1 and the first part of parent 2.

Fig. 5 One-point Crossover

- *Multiple-point crossover* (Fig. 6) – as in the explained above one-point crossover, but multiple crossover points are selected and everything in between the multiple points is swapped between the children.

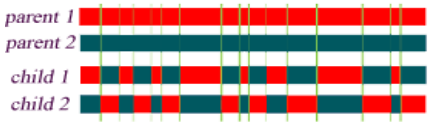


Fig. 6 Multiple-point Crossover

- *Uniform crossover* – there is ½ probability that newborn children will have 50% of the genes of each parent with randomly chosen crossover points. At each gene position, a random decision on whether the gene goes to the first child or the second is made.

2.5.5 Mutation

In genetics, a mutation is a change of the genome of an organism. A small part of the chromosome is altered.

Since genetic algorithms are local search algorithms, when a locally optimal point is found by a particular individual, the value of this individual's fitness function becomes the highest in the population. This individual manages to remain on top of the population for a number of iterations and all individuals start looking alike. To escape from this scenario a mutation function imposes variability in the population.

There are various mutation operators:

- *Replace* with new features mutation - with a small probability, a small part of the chromosome is replaced by a randomly generated new features.
- *Invert* mutation – a part of the chromosome is selected and the order of the features is inverted.
- *Insert* mutation – a small part of the chromosome is selected and inserted in another randomly chosen position of the chromosome.
- *Delete* mutation – a small part of the chromosome is selected and deleted. Therefore, the number of features decreases.
- *Swap* mutation – two small parts of the chromosome are selected and their elements are swapped.

3. Semi-supervised Multi-view Genetic Algorithm

3.1 Problem Statement

In this part a new approach to semi-supervised multi-view learning is proposed – *a semi-supervised multi-view genetic algorithm* (SSMVGA). This approach uses multiple sources of information, i.e. multiple views, and exploits both labeled and unlabeled examples. It minimizes the error (3.1).

$$Err(f_1,..,f_k) = \sum_{v=1}^{k}\left(\frac{1}{l}\sum_{i=1}^{l} c(x_i, y_i, f_v(x_i)) + \lambda\Omega_{SL}(f_v)\right) + \lambda_2 \sum_{u,v=1}^{k}\sum_{i=l+1}^{l+u} c(x_i, f_u(x_i), f_v(x_i)) \quad (3.1)$$

The error $Err(f_1,..,f_k)$ is the sum of the regularized empirical risk (based on the training set of labeled examples) of each view$_v$:

$$\frac{1}{l}\sum_{i=1}^{l} c(x_i, y_i, f_v(x_i)) + \lambda\Omega_{SL}(f_v)$$

plus the agreement among the views on the unlabeled examples:

$$\sum_{u,v=1}^{k}\sum_{i=l+1}^{l+u} c(x_i, f_u(x_i), f_v(x_i)).$$

The number of all view pairs is $C_k^2 = combination(k,2)$. The true values y_i of the unlabeled examples are not known. Therefore, for all unlabeled examples x_i and all pairs of views f_u, f_v - the predicted value of f_u is compared to the predicted value of f_v and the final disagreement between them penalizes the common error.

λ_1, λ_2 are the regularization parameters.

When $\lambda_2 = 0$, the algorithm ignores the unlabeled examples. This parameter defines the weight of the unlabeled examples, the impact of the agreement among the views.

Let $f_1^*,..,f_k^*$ be the learners for which *Err* ($f_1,...f_k$) is minimal:

$$f_1^*,..,f_k^* = \arg\min_{f_1,..,f_k} Err(f_1,..,f_k) \tag{3.2}$$

The proposed approach is a generalization of semi-supervised multi-view learning. It has the following strong points:

- It can be applied to multiple sources of data (not only two). Most of the approaches focus on two-view learning (co-training, co-EM, SVM-2K)
- It works for convex and non-convex functions. Approaches based on gradient descend require a convex function. When a function is not convex, it is a hard optimization problem. The error in (3.1) is not convex when:
  - the loss function is non-convex

    - **1/0 loss function**

    $$c(x_i, y_i, f(x_i)) = (y_i \neq f(x_i))$$
    $$c(x_i, y_i, f(x_i)) = \begin{cases} 1 & if\ y_i \neq f(x_i) \\ 0 & if\ y_i = f(x_i) \end{cases}$$

    - **hat function:**

    $$c(x, y, f(x)) = \max(1 - |w^T x_i + b|, 0))$$

    Semi-supervised support vector machines are based on this loss function (described in 2.3.4).
  - the regularizer (the penalty term) is a non-convex function [47]-capped l_1 norm ($\lambda \max(|w_i|, \theta)$, $\theta > 0$), SCAD penalty, MCP penalty [45].
- It does not require complex computations - reversed matrices, PCA (Principal Component Analysis), etc. A solution of a system of linear

equations does not always exist. It depends on the dataset and the rank of the matrix.

3.2 Methodology of the algorithm

The algorithm learns multiple hypotheses $f_1^*,\ldots, f_k^*$, simultaneously optimizing the error in (3.1). Each view is represented by a set of features $x^{(j)}$. $f_1^*,\ldots, f_k^*$ can be different functions (linear regression, logistic regression, polynomial regression, etc). $w_1,\ldots,w_k$ are the weight vectors corresponding to each view that should be learned. The number of features in each view can be different.

$$f_1^*(x^{(1)}) = f_1^*(x^{(1)}, w_1): \qquad w_1 = (w_{11},..,w_{1s})$$

...

$$f_k^*(x^{(k)}) = f_k^*(x^{(k)}, w_k): \qquad w_k = (w_{k1},..,w_{kp})$$

To learn $f_1^*,\ldots, f_k^*$, we have to learn the feature vectors $w_1,\ldots,w_k$. Genetic algorithms have already been successfully applied to neural networks parameter learning. The idea behind the proposed algorithm is to create a genetic algorithm, which optimizes $w_1,\ldots,w_k$ in the semi-supervised multi-view learning scenario.

3.2.1 Individuals

The individuals used in the semi-supervised multi-view learning genetic algorithm contain features from multiple views. The weights of the multiple views can be concatenated as if the algorithm were a single-view one.

Individual$_j$:

w_{11}	…	w_{1s}	…	w_{k1}	…	w_{kp}

Constructed in this way, the chromosome corresponds to k sub-chromosomes, which are concatenated. What is important is not to mix the features of the different views. Therefore, an individual can also be defined as:

Individual$_j$:

View 1	w_{11}	…	w_{1s}
View j	w_{j1}	…	w_{jl}
View k	w_{k1}	…	w_{kp}

3.2.2 Fitness Function

A better fitted individual has a higher fitness function and will contribute the most offspring to the next generation. Since the goal is to optimize the common regularized risk and find its minimum, a more fitted individual will then be defined as having a smaller value of *Err* (f_1,… f_k).

Fitness function:

$$fitness(p) = -Err(f_1,..,f_k) \tag{3.3}$$

3.2.3 Crossover

For crossover of individuals in the semi-supervised genetic algorithm *uniform crossover* was chosen and applied – with ½ probability newborn children have 50% of the genes of each parent with randomly chosen crossover points. Other crossover types may also be used provided that they do not change the size of the chromosome and do not mix the features of different views.

Individuals, whose fitness function is high are selected for reproduction with higher probability. The higher the value of the fitness function, the more times an individual is chosen to take part in crossover. Therefore, an individual, who has a high error value, is less likely to distribute genes in the new population.

$$\Pr(p) = \frac{fitness(p)}{\sum_{i=1}^{N} fitness(p_i)} \tag{3.4}$$

Pr is a function, which returns the probability of selecting an individual p to participate in crossover. N is the number of individuals in the population.

Selection is an important part of the algorithm as it also reflects the variety of the population. When only the best s% of the population participate in crossover, all

members of the population start looking alike. Mutation also plays an important part in sustaining decent diversity in the population.

3.2.4 Mutation

The chosen mutation function used in the semi-supervised multi-view genetic algorithm is "*replace with new features mutation"* – with a small probability a small part of the chromosome is replaced by randomly chosen new features. Since the individuals consist of weights - with a small probability, a small number of weights are replaced by randomly generated new ones. It is not sensible to shift features from one view to another, unless they are shifted within the views.

3.2.5 The Algorithm

At first, the algorithm generates a population of N individuals ($N = |P|$) of the type, described in 3.2.1.

It iterates MAX_ITER number of times, improving the population of individuals. MAX_ITER defines the stopping criterion of the algorithm and it stands for the maximum number of generations the algorithm will run. Other convergent techniques can also be applied. For example, evolve the population until there is no change in the fitness of the best S% of the population.

At each generation step:

- **crossover** of individuals is performed.
  The algorithm selects *t* pairs of parents (in accordance with the fitness function). The parameter *t* reflects the percentage of individuals that are going to be replaced at each iteration of the algorithm. Since at each crossover step two children are produced, the number of offspring is 2t.
- **Replaces** the worst individuals with the newly-generated children.
  2t number of individuals whose error is highest are excluded from the population. Due to their inaptitude, they do not survive.

- **mutation** - a small percentage of the population is altered. A mutation rate that is high may lead to loss of good solutions of the optimization problem.

Pseudocode:

```
void SemiSupervisedMultiViewGeneticAlgorithm()
{
        // Initialization – generate a population P of N individuals.
        // with randomly generated weights for each individual
        P = init(N);
        //Generations
        for(int i = 0; i < MAX_ITER; i++)
        {
                //generate t new pairs (offspring)
                for (int j = 1; j < t; j++)
                {
                        // selection – based on the fitness function
                        // defined in (3.3)
                        (parent1, parent2) = SELECTION(P);
                        (child1, child2) = CROSSOVER(parent1, parent2);
                        children.add(child1);
                        children.add(child2);
                }
                //Replace the worst individuals with the new offspring
                P` = ReplaceWorst(children);
                MUTATION(P`);
                P = P`
        }
}
```

3.3 Designing experiments

3.3.1 Experiments

One experiment is held comparing the following two genetic algorithms:

- supervised genetic algorithm (*Supervised GA*), which uses one view (concatenated features of the multiple views, the original set of attributes). No unlabeled examples are used.
- semi-supervised multi-view genetic algorithm (*Semi-supervised multi-view GA*) – the proposed algorithm, which relies on unlabeled examples and multiple sources of information.

The performance of the two algorithms is compared based on root mean squared error (RMSE). The aim of the comparison is to see whether using the proposed genetic algorithm (exploiting unlabeled examples and multiple views) reduces the error of the learned algorithm in comparison to a supervised equivalent, which uses a single view and no unlabeled examples.

3.3.2 Training and test sets:

In order to simulate a semi-supervised scenario and to evaluate the two algorithms, we need the labels of the unlabeled examples. Therefore, we need a fully-labeled dataset. From a fully-labeled dataset we can draw a random sample of examples - a fraction of labeled instances and "hide" the labels of the rest of the examples.

- The **training set** consists of:
  - *labeled examples* - a fraction of labeled examples from the original dataset. A small amount of examples is drawn at random and added to D_1.
  - *unlabeled examples.* The rest of the instances have the values of the regression function removed (hidden). These unlabeled examples are added to D_2.

A final training set is constructed (Fig. 7): $D = D_1 \cup D_2$.

original dataset

Examples	Attr 0	Attr 1	...	Attr K	Y
0	a00	a01	...	a0K	0
1	a10	a11	...	a1K	0
2	a20	a21	...	a2K	0
3	a30	a31	...	a3K	0
4	a40	a41	...	a4K	0
5	a50	a51	...	a5K	0
6	a60	a61	...	a6K	1
7	a70	a71	...	a7K	1
8	a80	a81	...	a8K	1
9	a90	a91	...	a9K	1
...	...	...	...	...	...
N	aN0	aN1	...	aNK	1

Examples	Attr 0	Attr 1	...	Attr K	Y
0	a00	a01	...	a0K	0
2	a20	a21	...	a2K	0
6	a60	a61	...	a6K	1
7	a70	a71	...	a7K	1

labeled

unlabeled

Examples	Attr 0	Attr 1	...	Attr K
1	a10	a11	...	a1K
3	a30	a31	...	a3K
4	a40	a41	...	a4K
5	a50	a51	...	a5K
8	a80	a81	...	a8K
9	a90	a91	...	a9K
...	...	...	...	...
N	aN0	aN1	...	aNK

Fig. 7 Train test – a random division

- The **test set** contains all the examples in D_2 (*transductive* semi-supervised learning). The performance of the algorithm will be evaluated based on these examples, using the original known labels (Fig. 8).

Examples	Attr 0	Attr 1	...	Attr K	Y
1	a10	a11	...	a1K	0
3	a30	a31	...	a3K	0
4	a40	a41	...	a4K	0
5	a50	a51	...	a5K	0
8	a80	a81	...	a8K	1
9	a90	a91	...	a9K	1
...	...	...	...	...	...
N	aN0	aN1	...	aNK	1

Fig. 8 Test set

3.3.3 Monte-Carlo Cross Validation

For error estimation Monte Carlo cross-validation [57] was used, the RMSEs of multiple random samples were averaged.

The dataset is randomly split into training and test sets. For each split, the two algorithms are run using the training set, and the error is assessed using the test set. Since the results depend on the initial labeled examples and their representativeness, multiple cross validation steps were performed and the final prediction is evaluated, based on the average of the observations.

A standard N-folded cross-validation can also be used (Fig. 9). Each fold corresponds to the amount of labeled examples. "One fold" number of examples is used as labeled instances and (N-1) folds are used as unlabeled instances. For example, if there are 1000 examples out of which only 10 are labeled, this leads to N = 100. If there are 10000 examples and only 10 labeled ones, N = 1000.

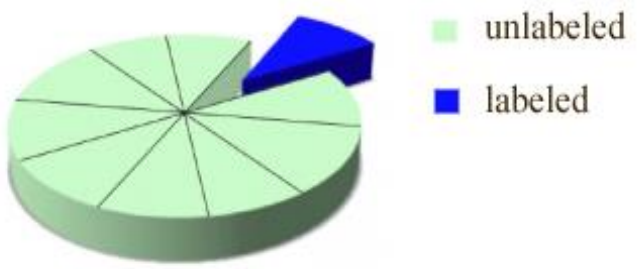


Fig. 9 Semi-supervised N-folded cross-validation

In order to compare two genetic algorithms, the following schema was used:

- For each cross validation step that is run both algorithms use the same training and test sets;
- The parameters of the two algorithms are the same (in terms of number of individuals, mutation rate, maximum number of iterations, etc.)
- For each cross validation step that is run both algorithms are executed multiple times, because genetic algorithms depend on the initially generated random values of the weights. As a result, only the best performing genetic algorithm is used in the future. This best genetic

algorithm (BGA) is chosen in accordance with the fitness function. As we have a small amount of labeled examples, we cannot afford to further divide them into a train and a validation set.

For each cross validation run, the RMSE of the BGA based on the test set (test RMSE - TRMSE) is calculated. TRMSE of all cross validation runs are averaged for the Supervised GA and the Semi-supervised multi-view GA.

3.4 Conducting experiments

3.4.1 Dataset

The performance of the semi-supervised multi-view genetic algorithm was evaluated based on a well-known dataset – "diabetes", which can be obtained from the UCI Machine Learning Repository [56]. The attributes of *diabetes* are real-valued, there are no missing values. It consists of 768 examples, 8 attributes and 2 classes. The two classes were treated as real values – 0.0 (tested negative) and 1.0 (tested positive) and the task was to learn a regression function.

3.4.2 Parameters of the algorithms

1. **Number of views**:

   k = 2 (Two views)

   The attributes were divided into the following two views:

   view_1= {preg, plas, pres, skin};

   view_2= {insu, mas, pedi, age};

2. **Type of regression functions**:

   Let f_1 and f_2 be two linear regression functions, whose vector features are w and v.

$$f_1(x) = w^T x_1$$
$$f_2(x) = v^T x_2 \qquad (3.5)$$

In linear regression, we can define the loss function as the squared loss function:

$$c(x, y, f(x)) = (y - f(x))^2 \tag{3.6}$$

3. Regularizer - a standard supervised regularizer for linear regression was used:

$$view\_1: \Omega_{SL}(f_1) = \| w \|^2 = \sum_{s=1}^{d} w_s^2$$

$$view\_2: \Omega_{SL}(f_2) = \| v \|^2 = \sum_{s=1}^{b} v_s^2 \tag{3.7}$$

Therefore, we have to find such a tuple (f_1^*, f_2^*):

$$(f_1^*, f_2^*) = \arg\min_{f_1, f_2} \frac{1}{l} \sum_{i=1}^{l} ((y_i - \mathrm{w}^{\mathrm{T}} \mathrm{x}_{1,\mathrm{i}})^2 + (y_i - \mathrm{v}^{\mathrm{T}} \mathrm{x}_{2,\mathrm{i}})^2) + \lambda_1 \| w \|^2 + \lambda_1 \| v \|^2 + \lambda_2 \sum_{i=l+1}^{l+u} (\mathrm{w}^{\mathrm{T}} \mathrm{x}_{1.\mathrm{i}} - \mathrm{v}^{\mathrm{T}} \mathrm{x}_{2,\mathrm{i}}) \tag{3.8}$$

This problem is also known as ridge regression [27].

4. **Maximum number of iterations**:

   MAX_ITER = 20000

5. **Regularization parameters**:

   - selected values: $\lambda_1 = 0.5, \lambda_2 = 0.05$

   It was experimented with different values for the two regularization parameters and those that have the smallest error were chosen.

6. **Number of individuals**:

   N = 100

7. **Mutation rate**: 5%.

   Out of the 100 individuals only 5 are mutated.

3.4.3 Results

In Table 1 the comparison between the supervised genetic algorithm (Supervised GA) and the proposed semi-supervised multi-view genetic algorithm (Semi-supervised multi-view GA) can be seen. The number of labeled examples is 20. The rest 748 examples are used as unlabeled. The two algorithms are compared in

accordance with TRMSE. The semi-supervised multi-view genetic algorithm outperforms its supervised equivalent as it has: TRMSE = 0.63, whereas the TRMSE of its supervised equivalent is 0.81.

Table 1 Semi-supervised multi-view GA vs Supervised GA

Algorithm	TRMSE
Supervised GA	0.81
Semi-supervised multi-view GA	0.63

Table 2 contains the results of the same experiment, but 20% of the unlabeled examples are excluded from the training set and left for a test set (*inductive* semi-supervised leaning). The same number of labeled examples is used. 37-folded cross-validation is performed. The aim is to explore the performance of the algorithm on future unobserved examples which are not used as unlabeled examples in the training set. It does not affect the performance of the supervised genetic algorithm as it does not use unlabeled examples for learning. The error of the semi-supervised multi-view genetic algorithm after the process of cross-validation is 0.59 and is close to the Monte Carlo one (Table 1) and shows that we can use the semi-supervised multi-view genetic algorithm to predict the labels of instances which are outside the set of unlabeled training examples.

Table 2 Semi-supervised multi-view GA: inductive learning

Algorithm	*TRMSE*
Semi-supervised multi-view GA	0.59

3.5 Conclusion

Darwin's concept of evolution has been carried over into genetic algorithms for solving some of the most demanding problems for computer optimization. In this section, such a minimization problem has been solved with the aid of a genetic

algorithm. In order to minimize the common error which depends on *k* views, a semi-supervised multi-view genetic algorithm was proposed. An individual has *k* sub-chromosomes, which contain features from *k* data sources. The proposed algorithm uses both labeled and unlabeled examples. The algorithm improves the population of individuals, striving after such learners, which agree with each other on the unlabeled examples. The experimental results prove that using multiple views and unlabeled examples reduces the error of the algorithm.

The aim of the experiment was to "simulate" a semi-supervised framework by using only a fraction of the examples as labeled. In the semi-supervised scenario the performance of the proposed algorithm is evaluated and compared to a supervised one. The results from the previous section inspired the application of the semi-supervised multi-view genetic algorithm to sentiment analysis in Bulgarian (Part 4) – a field where labeled examples in Bulgarian are hard to find and require many experts to classify the examples beforehand.

4. A Sentiment Analysis System in Bulgarian

Humans can feel emotions, express their feelings and understand each other relying on hidden patterns of sentiment. We can formulate a simple sentence as a neutral statement, but we can also impose some emotions in it, which makes it an opinion.

Sentiment analysis [80, 81, 83, 91] is a modern discipline, which is responsible for extracting subjective information, understanding the emotions encoded in the text given. It uses a variety of algorithms in order to achieve this. Being an interdisciplinary subject, combining algorithms of artificial intelligence, natural language processing, information retrieval and statistics, it simulates intelligent behavior.

Social media (Facebook, Twitter, Google+, LinkedIn) contains large amounts of data – comments and articles, which express personal opinion. For examples, it may be of interest to know what the opinion of social media on a certain topic is - is it positive or negative [93, 94]? Is a product positively accepted by the customers? Does a politician have a positive public image? Is a football couch popular among the fans? These are all important questions, whose answers are valuable for many contemporary businesses. To answer these questions, we need to develop tools for automatic extraction of sentiment from texts.

4.1 Sentiment Analysis in English

There has been considerable research in the field of sentiment analysis in English (a.k.a. opinion mining), it has grown rapidly to become one of the most actively researched areas in NLP (natural language processing). Modern directions in sentiment analysis are:

- *classification of positive/negative/neutral reviews*;

- *classification of personality traits* - (nervous, anxious, reckless, morose, hostile, jealous);
- *classification of emotions* - angry, sad, joyful, fearful, ashamed, proud, elated;
- *detecting irony* [95]*;*
- *rumor credibility* in documents;
- *plagiarism;*
- *author recognition.*

Most of the research has been conducted in English. Bing Liu [21] overviews the state of art in different types of sentiment analysis and discusses a wide variety of practical applications. Kennedy and Inkpen [22] present two methods for determining the sentiment expressed in movie reviews in English. The first method is based on a sentiment lexicon [23, 50] and relies on the polarity of the very words in the documents. Usually, we sum the weights of the negative and positive words and a final decision is made based on the outcome. The second approach uses a classification algorithm in order to differentiate between the classes. The current dissertation focuses on sentiment analysis, based on a machine learning classification/regression algorithm.

It should also be noted that an opinion is generally subjective. Two people might not agree on the sentiment of a sentence. Furthermore, some sentiment analysis competitions (Semantic Evaluation – "SemEval") remove the reviews the judges could not agree on from the dataset.

4.2 Sentiment analysis in Bulgarian

Sentiment analysis in Bulgarian (like in Chinese [6]) suffers from labeled examples shortage. There is little labeled data, which can be used in the learning process. It is both scarce and incomplete. In such cases, when classified examples are hard to obtain, to improve the classification accuracy of the algorithm unlabeled examples are collected so that relevant information can be extracted from them.

Furthermore, multiple sources of data are explored and semi-supervised multi-view learning techniques are applied.

In order to cope with the burdens of insufficient data in Bulgarian, a new learning system for sentiment analysis needs to be developed in the Bulgarian language. This thesis proposes a working model of such a system.

The sentiment analysis system for the Bulgarian language (**SASBG**) is an innovative framework, which predicts the sentiment of movie reviews. It is a semi-supervised sentiment analysis system and has the following characteristics:

- The system uses multiple sources of information. Since there is little labeled data in Bulgarian, a second English data source is also used. Each instance has attributes from multiple sources of data (a Bulgarian and English view);
- The system uses labeled English movie reviews and unlabeled Bulgarian movie review. It benefits from the unlabeled examples;
- The system is based on a new semi-supervised multi-view learning approach - the algorithm proposed in part 3.
- The system can process large amounts of information (big data). It has a distributed framework and can process parallel calculations.

4.3 SASBG Development Cycle

The development of the learning system for sentiment analysis in Bulgarian includes the following phases:

- **Data understanding and data acquisition** – a process of greatest importance, which is responsible for understanding what a review is, why we need a new system, where we can find data for it;
- **Data preprocessing** – preprocessing the originally obtained data, so that the reviews are ready for the next phase;
- **Data translation** – since multiple sources of information are used, we need to manually create the second view by translating the reviews.

The English reviews are translated into Bulgarian with Google translate and the Bulgarian movie reviews are translated into English;

- **Feature extraction** – with the aid of Gate features are extracted from the two views;
- **Modelling** - development of a distributed framework: a cluster of machines running under Linux Mint. The software architecture uses Apache Hadoop (*hdfs* and *yarn*) and Apache Spark;
- **Evaluation** of the semi-supervised sentiment analysis system. Multiple experiments are carried out. The semi-supervised multi-view genetic algorithm is compared to its supervised equivalents.

Usually, data translation and feature extraction are part of the data preprocessing phase, but because they required too much effort and also in view of the better understanding of the system, they were left as separate stages in the cycle. The pipeline of the data mining cycle can be seen in Fig. 10. As displayed, the data mining cycle supports moving back and forth between different phases. The information learned during one process can trigger new questions and ideas. Therefore, going back to previous phases is paramount.

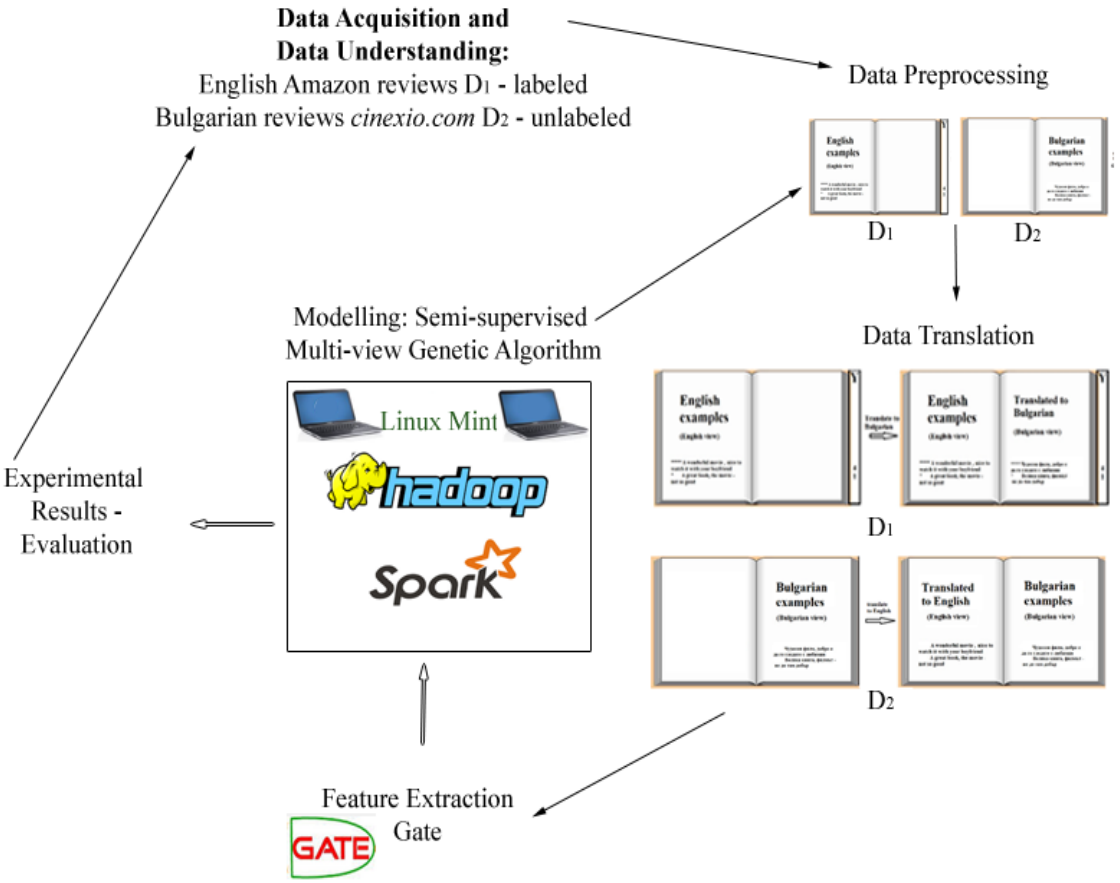


Fig. 10. Sentiment Analysis in Bulgarian – data mining cycle

4.4 Movie reviews - data acquisition and data understanding

The difficulties of sentiment analysis in Bulgarian lie in the absence of labeled Bulgarian training sets. An expert in the field should label all training examples manually. Unlike a labeled set of instances, an unlabeled one is easy to find. Most of the web-based systems and forums provide us with "a comment functionality". You can comment on a product and point out its strong sides or shortcomings. However, you cannot rate your own opinion, as either positive or negative. Even when a person comments on an article in the social networks (Facebook, LinkedIn), it does not say anything about the sentiment of the comment,

unless the system teaches itself to forecast it. These comments represent unlabeled examples.

There are various abundant sources of labeled datasets for sentiment analysis in English. Some of the most widely used are:

- http://www.tripadvisor.com/ – contains numerous labeled reviews. There also are:
  - http://www.tripadvisor.ru/ (in Russian)
  - http://www.tripadvisor.jp/ (in Japanese), etc.;

  In Bulgarian there is no such counterpart.
- http://www.booking.com/ - contains numerous labeled English reviews.
- http://www.amazon.com – contains numerous labeled English reviews in multiple domains.
- http://www.ebay.com – contains numerous labeled English reviews in different domains.
- http://www.imdb.com/ - contains numerous labeled English movie reviews.
- Twitter sentiment corpuses;

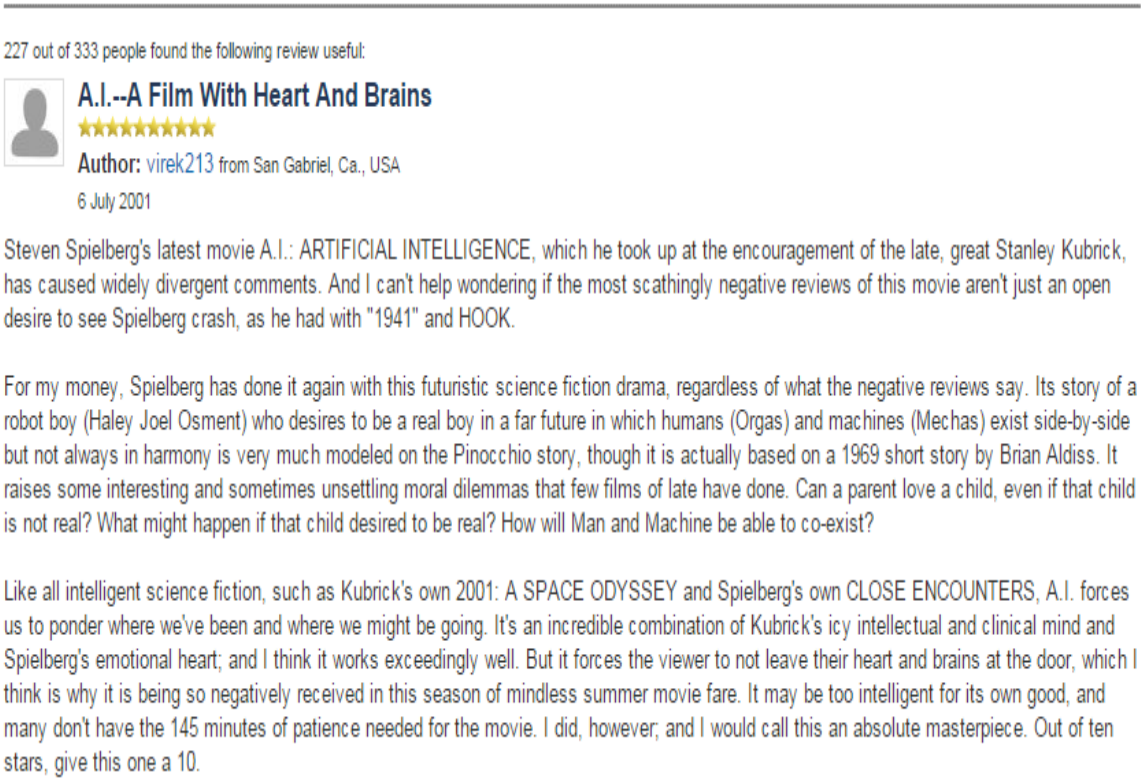
In this thesis, *amazon* movie reviews [30] are used.

The proposed system is in the field of natural language processing. It uses texts as data (each text or review is an example). Based on previous history (a training set), it learns to forecast the sentiment of a new document. The domain of the texts can vary. For the experimental results, the field of movie reviews was selected. It was chosen because there is current interest in the field and a growing need for such a learning system in Bulgarian.

4.4.1 Movie reviews

A movie review consists of sentences, expressing the user's attitude towards a movie. The user assesses the quality of the film, based on a scale. For example, the

review from Fig. 11 was obtained from *http://www.imdb.com/* and expresses the opinion of a user on the movie "A.I. Artificial Intelligence", (2001). The user must have liked the movie very much giving it a score - 10/10 (**********).



227 out of 333 people found the following review useful:

A.I.--A Film With Heart And Brains

★★★★★★★★★★

Author: virek213 from San Gabriel, Ca., USA

6 July 2001

Steven Spielberg's latest movie A.I.: ARTIFICIAL INTELLIGENCE, which he took up at the encouragement of the late, great Stanley Kubrick, has caused widely divergent comments. And I can't help wondering if the most scathingly negative reviews of this movie aren't just an open desire to see Spielberg crash, as he had with "1941" and HOOK.

For my money, Spielberg has done it again with this futuristic science fiction drama, regardless of what the negative reviews say. Its story of a robot boy (Haley Joel Osment) who desires to be a real boy in a far future in which humans (Orgas) and machines (Mechas) exist side-by-side but not always in harmony is very much modeled on the Pinocchio story, though it is actually based on a 1969 short story by Brian Aldiss. It raises some interesting and sometimes unsettling moral dilemmas that few films of late have done. Can a parent love a child, even if that child is not real? What might happen if that child desired to be real? How will Man and Machine be able to co-exist?

Like all intelligent science fiction, such as Kubrick's own 2001: A SPACE ODYSSEY and Spielberg's own CLOSE ENCOUNTERS, A.I. forces us to ponder where we've been and where we might be going. It's an incredible combination of Kubrick's icy intellectual and clinical mind and Spielberg's emotional heart; and I think it works exceedingly well. But it forces the viewer to not leave their heart and brains at the door, which I think is why it is being so negatively received in this season of mindless summer movie fare. It may be too intelligent for its own good, and many don't have the 145 minutes of patience needed for the movie. I did, however; and I would call this an absolute masterpiece. Out of ten stars, give this one a 10.

Fig. 11. Review of the movie "A.I. Artificial Intelligence"

4.4.2 Rating scale

Usually, in sentiment analysis, an instance is treated either as "positive" or "negative". A "neutral" class can also be used. The movie reviews used in this thesis, have a rating scale varying from 1 to 5 (*, **, ***, ****, *****). The rating of the movies is treated as a real-valued function.

4.4.3 Multiple sources of information

Two sources of data are used - Bulgarian reviews (unlabeled examples) and English movie reviews (labeled examples).

English reviews:

As explained above, an English source of classified instances is easy to find and use. 992 English Amazon movie reviews [30] were used. These reviews are rich in content. They will be translated into Bulgarian.

Bulgarian reviews:

Since automatic machine translation into Bulgarian is not that good, a second source of information needs to be used, i.e. genuine Bulgarian reviews. A set of movie reviews in Bulgarian was manually extracted from *www.cinexio.com*. It consists of 100 labeled examples and they were translated into English. These examples were used as unlabeled, the labels were "hidden" during the learning phase. The labels were used only for the evaluation of the results.

4.5 Data Preprocessing

Data preparation is an important phase that can have a major impact on the final results. Analyzing data that has not been carefully preprocessed can produce misleading results.

The Amazon movie reviews did not contain missing information and were preprocessed by the author who published them.

The Bulgarian reviews extracted from *cinexio.com* were not that good and needed to be preprocessed. Some of them were removed from the training set because of duplication or because they were not in Cyrillic. Some spelling mistakes were manually corrected.

4.6 Data Translation - creating two views

We need two views for both the English and Bulgarian reviews to be further used in the modelling phase. Data translation plays a major role in the system and is a phase that cannot be omitted. Translation was executed with the help of Google translate.

English reviews

The *amazon* reviews were translated into Bulgarian by Google translate so that each review has two views (Fig. 12). These examples form the training set of labeled examples D_1.

x_i = *(original_English_view, translated_into_Bulgarian_view)* ,

y_i = *Rating of* x_i

$$D_1 = \{(x_i, y_i)\}_{i=1}^{l}$$

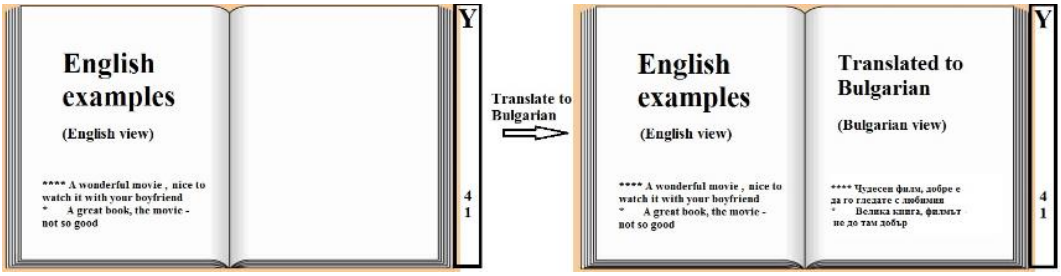


Fig. 12 Labeled examples

Bulgarian reviews

The Bulgarian reviews were translated into English so that a second view is formed (Fig. 13). As the ratings were "hidden", these reviews are added to the pool of examples as unlabeled examples D_2.

x_j = *(translated_into_English_view, original_Bulgarian_view)*

$$D_2 = \{x_j\}_{j=l+1}^{l+u}$$

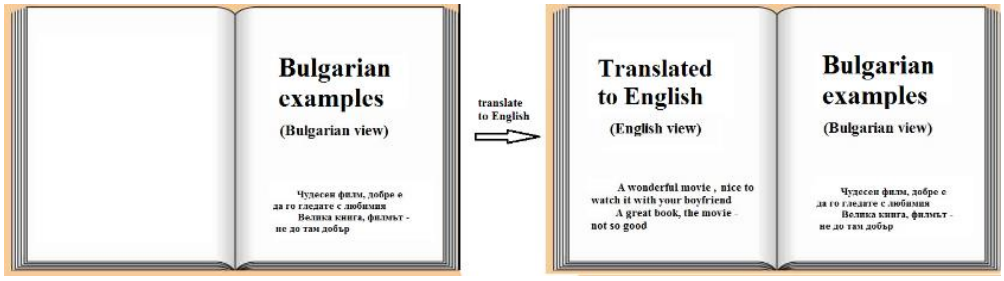


Fig. 13 Unlabeled examples

4.6.1 Feature Extraction from movie reviews

After the data preprocessing and data translation phases, cleansed and compatible .txt files were produced. They were further used by the feature extraction module so that a vector of feature weights could be extracted.

Natural language processing (NLP) enables computers to extract meaning from natural language texts. The "meaning" in this dissertation refers to sentiment and the intelligence to learn to predict it based on a text.

Each movie review can be viewed as a separate document, a text or a bag of words. It consists of sentences and depending on the content, the very words inside, it expresses some sentiment. Therefore, each movie review is viewed as an example. A collection of movie reviews (documents) forms a dataset.

Movie review/document j:

$$d_j = (w_{j0}, ..., w_{jm})$$

A movie review is defined by a vector of weights, one component for each term in a dictionary of words. The corpus of movie reviews gives us a matrix, which can also be viewed as a vector space [28].

Terms

A term can be a single word or a longer phrase. In SASBG, biwords were used. A biword index is an approach to handling phrases, which consider every pair of consecutive terms in a document as a phrase. For example, the text "Lovely morning sunshine" will generate the following biwords: "lovely morning" and "morning sunshine". If words were chosen for terms, the dimensionality of the review vector space would be the number of words in the vocabulary - the number of distinct words, which occur in the corpus. As long as the biword model was chosen, the dimensionality rose considerably.

4.6.2 Scoring the movie reviews

The weight of each term follows the *tf-idf* standard:

Term frequency - the number of times that term *t* occurs in document *d*: $tf(t,d)$;

Inverse document frequency - it measures whether the term is common or rare across all documents (N is the number of documents in *D, t* is the term):

$$idf(t, D) = \log \frac{N}{|\{d \in D, t \in d\}|} \tag{3.8}$$

Weights:

$$w_t = tfidf(t, D) = tf(t, d) * idf(t, D) \tag{3.9}$$

4.6.3 Gate pipeline

Gate [29] is a NLP framework, solving text processing problems.

In order to extract features from the Bulgarian and English views the following pipeline was created (Fig. 14):

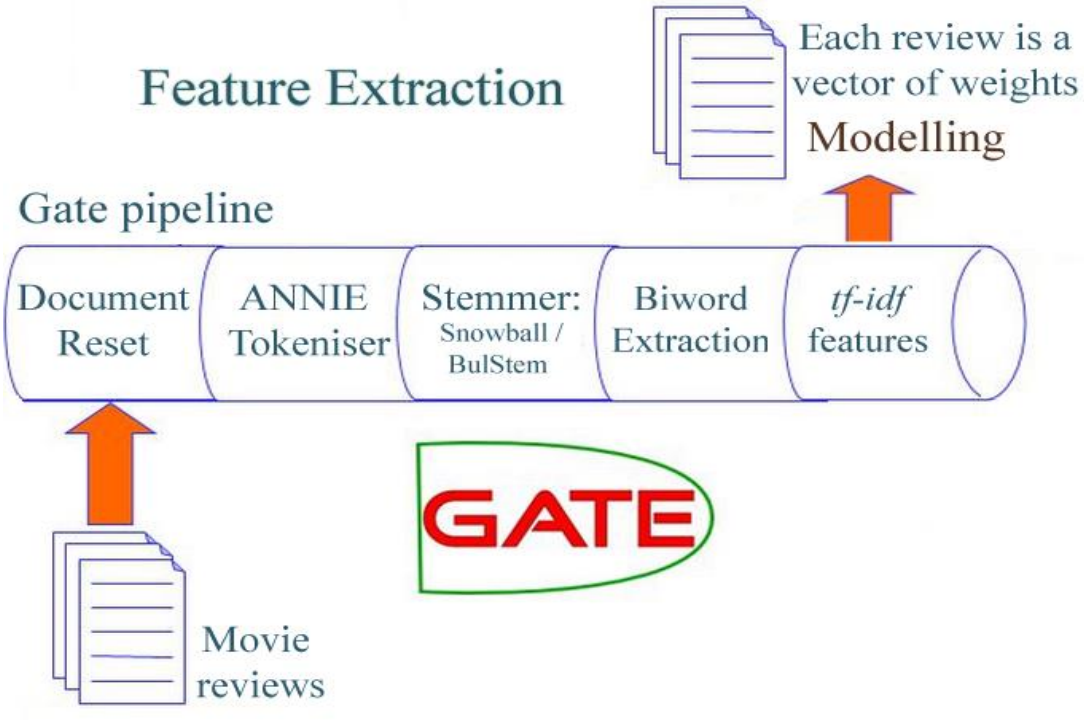


Fig. 14. Feature extraction

The feature extraction process includes the following steps:

- *Document Reset* – it enables the document to be reset to its original state, by removing all the annotation sets and their contents;
- *ANNIE Tokeniser* – splits the text into very simple tokens;
- *Stemmer* - reduces derived words to their word stem or root form;

- *Snowball stemmer* – an English stemmer
- *BulStem* – a Bulgarian stemmer

- *Groovy Script* – a groovy script was written, which is used for:
  - biword feature extraction – a biword index is created. The groovy script calculates the number of times a term appears in each review and the number of documents it appears in.
  - The tf-idf weights are calculated and a final output file is returned, containing one record (a vector of weights) for each movie review.

Since two sources of information were used, the feature extraction process was done once for the Bulgarian and once for the English view.

Extracted features:

- Bulgarian view: 17099 features were extracted
- English view: 12391 features were extracted

4.7 Modelling of SASBG

In modelling, data mining algorithms are built - manually implemented or existing libraries and tools are applied. This stage is the implementation of the software architecture framework. A distributed framework, which can process big data is designed and developed. Implementation of the *semi-supervised multi-view genetic algorithm* is also included.

4.7.1 Distributed Software Architecture

High-Speed In-Memory Analytics

In-memory data storage is very important for the fast execution of algorithms. It is significantly valuable, especially, for iterative procedures like genetic algorithms, which process the same dataset again and again. A distributed system gives us much more memory capacity (a cluster of nodes can provide us with almost limitless resources, depending on the number of nodes).

In the previous section, it can be seen that 17099 features were extracted from the Bulgarian view and 12391 – from the English.

The memory storage space required was calculated to be:

- *Bulgarian view*: 17099 features*100 examples*8 bytes: ~13MB
- *English view*: ~12391 features *1000 examples *8 bytes: ~95MB.

For future works skip-words or more complex indices (currently biwords) can be explored. There might also be more than two views. There can be more examples – both labeled and unlabeled. Therefore, the in-memory estimation can rise considerably.

Parallel Execution of calculations

Another advantage of a distributed framework is that it executes parallel calculations on different nodes.

MapReduce is a programming model, processing large data sets in parallel, performing two steps:

- *Map Step* – a master node assigns calculations to worker nodes, calling the same operators on different examples from the dataset. A map function is defined by the user, the same rules are applied to all instances. It produces a key/value pairs.
- *Reduce Step* – a master node assigns calculations to worker nodes. The pairs with equal keys are combined based on a defined *reduce* function and a single result is returned for each key.

MapReduce example:

We can use the map/reduce paradigm to calculate the following sum:

$$\sum_{i=1}^{l}((y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_{1,\mathrm{i}})^2 + (y_i - \mathbf{v}^{\mathrm{T}}\mathbf{x}_{2,\mathrm{i}})^2)$$

Python (Apache Spark) code:

```
error = labeled.map(
              lambda x:
                     (x[0] - x[1: 1 + VIEW_1].dot(w1[i]))**2 +
                     (x[0] - x[1 + VIEW_1:].dot(w2[i]))**2
        ).reduce(add)
```

For each example in "`labeled`", $(y_i - \mathrm{w}^{\mathrm{T}}\mathrm{x}_{1,i})^2 + (y_i - \mathrm{v}^{\mathrm{T}}\mathrm{x}_{2,i})^2$ is calculated and the sum of the elements is returned via a reduce function.

- `x[0]` is the y_i value of the example x_i,
- `x[1: 1 + VIEW_1]` is the vector of the attributes of view 1,
- `x[1 + VIEW_1:]` is the vector of the attributes of view 2,
- `dot()` performs a dot multiplication of two vectors.

How big is "Big Data"? When should it be used?

Big data implies large amount of data, which cannot be processed by a single machine and its resources. Not only cannot it be stored physically, but there is no execution power to afford the learning and prediction processes. Whether to use or not a distributed system depends on the task and resources we have. If we can process the task with the aid of one machine in reasonable time, there is no need of it. But if we cannot process it or it takes longer time, a distributed framework is preferred.

The system architecture of SASBG can be seen In Fig. 15.

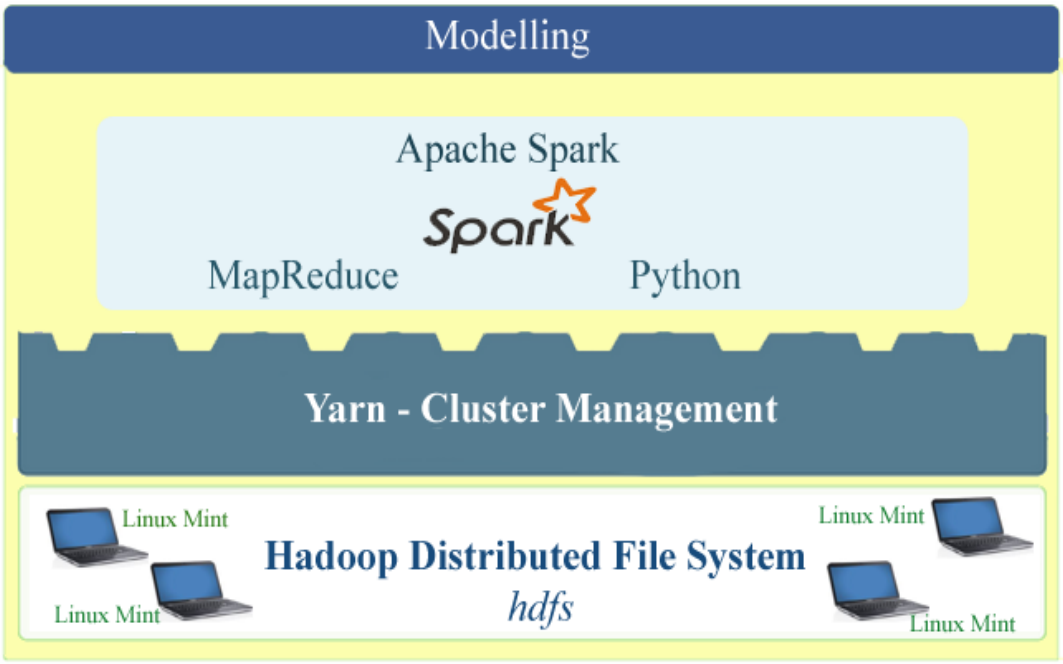


Fig. 15 SASBG Architecture

The hardware architecture comprises a cluster of two laptops, running under Linux Mint (a distribution based on Debian).

4.7.2 Apache Hadoop

Apache Hadoop [31] is an open-source software framework, which was written in Java for distributed storage and distributed processing of very large data sets on a cluster of computer nodes. Recently, it has gained popularity and turned out to be applicable in such areas as NLP. The hadoop distributed file system (*hdfs*) as well as its cluster management – *yarn* are penetrating important business fields worldwide, which makes them stand out as crucial.

Although Apache Hadoop has its own MapReduce engine, it was not used in the modelling process. Only the *hdfs* file system and the cluster manager *yarn* were used.

4.7.3 Apache Spark

Apache Spark [32] is an open-source cluster computing framework, which provides performance up to 100 times faster than hadoop's MapReduce. It has a machine learning module MLlib and multiple MapReduce examples. Due to the good documentation and examples, personal code is easy to write.

Apache Spark was chosen for the software architecture because of its applicability and usability - it is fast and easy to learn. It runs on *hdfs* and uses *yarn*. Spark supports three programming languages: scala, java and python. For the implementation of the algorithms python was chosen.

Python further includes a library called "NumPy" - an extension to the Python programming language. This library adds support for large, multi-dimensional arrays and matrices. It also provides high-level mathematical functions to operate on these arrays.

4.8 Evaluation of SASBG

In this section, the performance of SASBG is evaluated. The system is also compared to supervised equivalents.

4.8.1 Training and test sets

The training and test sets contain English and Bulgarian reviews after the processes of data preprocessing, data translation and feature extraction.

- The *training set* contains both labeled and unlabeled examples

$$D = D_1 \cup D_2.$$

These are the labeled English reviews and the unlabeled Bulgarian reviews.

- The *test set* contains all the examples in D_2 (the Bulgarian reviews). We will compare the algorithm to its supervised equivalents based on these examples (using the original known labels). D_2 contains only unlabeled examples. Cross-validation will not affect the supervised algorithms at all - they use only labeled examples.

  The distribution of labels in the test can be seen in Fig. 16. Out of the 100 examples:

  - 43 have a rating *****;
  - 20 have a rating ****;
  - 14 have a rating ***;
  - 8 have a rating **;
  - 15 have a rating *.

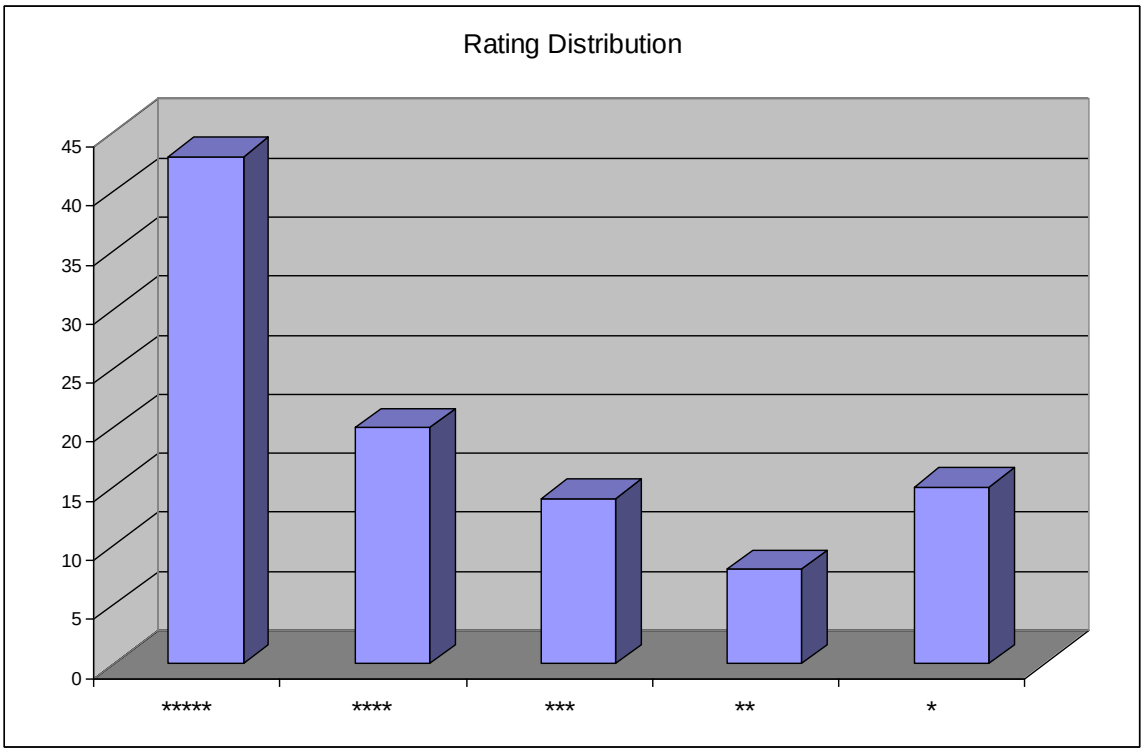


Fig. 16 Rating Distribution

4.8.2 Designing Experiments

One experiment is performed comparing 5 methods - SSMVGA is compared to four supervised genetic algorithms. All genetic algorithms use the same set of labeled examples, the same parameters – number of iterations, number of individuals in the population, and mutation rate. Each genetic algorithm is executed multiple times and the best performing is chosen for final comparison. Detailed information can be seen in Table 3.

Table 3 Sentiment Analysis - compared algorithms

Algorithm	Bulgarian view	English view	Labeled examples	Unlabeled examples
Supervised 1-view (*SGA)*	V	V	V	
Supervised 2-view (*SMVGA)*	V	V	V	
Supervised – Bulgarian (*BSGA)*	V		V	
Supervised – English (*ESGA)*		V	V	
Semi-supervised 2-view (*SSMVGA)*	V	V	V	V

The following 5 genetic algorithms are compared:

- ***Supervised*** – *English* (English supervised single-view genetic algorithm - *ESGA*): only the English view is used. For the learning phase the original English reviews are directly used and for the prediction one - the Bulgarian test set is translated into English and tested. No unlabeled examples are used.
- ***Supervised*** – *Bulgarian* (Bulgarian supervised single-view genetic algorithm - *BSGA*) – only the Bulgarian view is used. For the learning phase the original English reviews are translated into Bulgarian and then used, for the prediction one - the Bulgarian test set is directly tested. No unlabeled examples are used.
- ***Supervised*** *1-view* (supervised single-view genetic algorithm - *SGA*) – the features of the two views are concatenated in one view and used for learning and prediction. No unlabeled examples are used.
- ***Supervised*** *2-view* (supervised multi-view genetic algorithm - *SMVGA*) – both the English and the Bulgarian views are used for learning and prediction, no unlabeled examples used.
- ***Semi-supervised*** *2-view* (semi-supervised multi-view genetic algorithm - *SSMVGA*) – both the English and the Bulgarian views are

used for learning and prediction, unlabeled examples are used as well. This is the algorithm proposed in this paper.

4.8.3 Conducting Experiments

The experimental results are displayed in Table 4. It can be seen that the English supervised GA is the weakest. The reason might lie in poor translation from Bulgarian into English. It can be seen that it worsens the prediction of both the Supervised 1-view and Supervised 2-view GAs. The multi-view supervised 2-view GA (RMSE = 2.84) performs better than its 1-view counterpart (RMSE=3.09). They are both better than the single English GA (RMSE=3.12).

The 2-view semi-supervised GA (RMSE = 2.16) manages to outperform the single Bulgarian GA, because it also relies on unlabeled examples and multiple views.

The five algorithms are also compared based on MAE (Mean Absolute Error). The results are displayed in Table 4.

Table 4 Sentiment Analysis – experimental results

Algorithm	RMSE	MAE
Supervised 1-view (*SGA)*	3.09	2.20
Supervised 2-view (*SMVGA)*	2.84	2.10
Supervised – Bulgarian (*BSGA)*	2.20	1.90
Supervised – English (*ESGA)*	3.12	2.22
Semi-supervised 2-view (*SSMVGA*)	**2.16**	**1.86**

The parameters of all genetic algorithms are:

- Maximum number of iteration (stop criterion)

  $\mathrm{MAX\_ITER} = 10000$

- regularization parameters

  $\lambda_1 = 0.5, \lambda_2 = 0.5$

  It was experimented with different values for the two regularization parameters and such were chosen that lead to the smallest error.

- Number of individuals in the population

  $N = 100$

- $\text{Mutationrate}: 5\%$

  Out of the 100 individuals only 5 are mutated.

5. A Multi-View teaching algorithm

5.1 Problem Statement

The *multi-view teaching algorithm* (MTA) is a modification of the standard co-training algorithm. Blum and Mitchell [3, 45] published the co-training assumptions:

- the algorithm requires two strong classifiers;
- the two views should be conditionally independent given the class:

$$P(X_1 | Y, X_2) = P(X_1 | Y)$$

$$P(X_2 | Y, X_2) = P(X_2 | Y)$$

What happens when one of the classifiers is a bad learner, i.e. weaker than the other? When one of the classifiers is not a good learner, its predictions are expected to be false, it augments the set of labeled examples of the other learner with instances, which are misleading. As a result, it worsens the final prediction.

The proposed modification takes this into consideration and tries to improve only the weaker classifier. The multi-view teaching algorithm increases only the list of labeled examples of the weaker classifier, based on the most confident examples of the other learner.

5.2 Methodology of the algorithm

The multi-view teaching algorithm is a semi-supervised two-view learning algorithm. It uses unlabeled examples, which have projections on two sources of information. On each view a separate classifier is learned L_j. The final prediction of the algorithm is a combination of the two views.

Let L_1 and L_2 be two Bayes classifiers (*naïve Bayes classifiers or Bayes classifiers, based on multivariate normal distribution*).

θ_1 - the parameter of L_1.

θ_2 - the parameter of L_2.

- When the underlying classifiers L_1 and L_2 are two naive Bayes classifiers with real-valued attributes:

$$\theta = (\mu, \sigma)$$

- When the underlying classifiers are two Bayes classifier, based on multivariate normal distribution with real-valued attributes:

$$\theta = (\pi, \mu, \Sigma)$$

Let L_2 be the weaker learner, with worse classification accuracy. Let U_2 be the set of labeled examples of classifier L_2.

W contains all unlabeled examples.

At first, the algorithm learns L_1 and finds its parameter θ_1. L_1's most confident examples of each class are added to L_2's pool of labeled examples so that we can find θ_2.

Algorithm:

```
void multiViewTeachingAlgorithm()
{
    // U1 contains the attributes of view1 (only labeled examples)
    U1 = view1(D1);
    // U2 – contains the attributes of view2 (only labeled examples)
    U2 = view2(D1);
    W = D2; // all the unlabeled examples;
    θ1 = learn(L1, U1);  //learn θ1 based on view1
    // Add more labeled examples to  L2
    for (xi in W)
    {
        for(int j = 1; j < K; j++) // For each class yj:
        {
            y[j] = P(yj | xi, θ1);
```

```
            }
            // y* is the maximal element of the array y[]
            y* = max(y);
            list.add(xi, y*);
        }
        for (int j = 1; j < K; j++) // For each class yj:
        {
            // find the most confident examples and if they exceed some
            threshold add them to U2
            U2 += mostConfident(j);
        }
        θ 2 = learn(L2, U2);  //based on view2
    }
```

The set of unlabeled instances (W) can be further divided into bins and the bins can be processed in parallel. This optimization technique will be extremely useful for datasets with lots of unlabeled examples.

Another point worth mentioning, is that the algorithm tries to find the most credible examples of L_1. Not all newly labeled examples are added to the pool of labeled examples of the second classifier L_2. If the fitness of an example does not exceed some threshold, it is not used. If the first learner is not confident about some unlabeled example, this example is not used further in order not to worsen the second learner L_2. It was experimented with multiple thresholds and the chosen one was selected experimentally - the one that brings the highest classification accuracy.

The algorithm is not an iterative procedure (the original co-training algorithm is), which makes it suitable for image/movie processing, tracking objects, etc. When considering a semi-supervised framework, where the number of labeled examples is small, every new representative example is important.

After the correction of the weaker classifier, it can be proceeded with an iterative co-training procedure.

Combining the views

The results of the two classifiers are intuitively combined so that a final decision about the classification of new examples is made. Blum and Mitchell proposed a simple framework where the final class probability of a new example is the multiplication of the two views' probabilities:

$$P(y_j \mid x_i) = P(y_j \mid x_i, \theta_1) P(y_j \mid x_i, \theta_2) \tag{5.1}$$

To escape from the multiplication in (5.1), a standard log-procedure was used, the argmax remains the same:

$$\begin{aligned} &\arg\max_{y_j} P(y_j \mid x_i, \theta_1) P(y_j \mid x_i, \theta_2) = \\ &\arg\max_{y_j} \log P(y_j \mid x_i, \theta_1) + \log P(y_j \mid x_i, \theta_2) \end{aligned} \tag{5.2}$$

5.3 Designing experiments

5.3.1 Experiments

One experiment is held and the performance of the following algorithms is compared:

- multi-view teaching algorithm (MVTA) based on two naïve Bayes classifiers
- supervised naive Bayes classifier (NB).

The aim of the experiment is to determine whether the proposed algorithm, which uses unlabeled examples and multiple views leads to a smaller error compared to that of a standard supervised method.

5.3.2 Training and test sets:

A Monte-Carlo cross-validation was used; multiple random samples were averaged. To reduce variability, multiple rounds of cross-validation steps were performed using different partitions, and the validation results were averaged over the rounds.

At each Monte Carlo cross-validation step the training and test sets are constructed in the following way:

- *Training set*
  - *Labeled examples* - The training set consists of a *fraction* of the examples from the original dataset.
  - *Unlabeled examples* - The rest of the instances have their classifications removed. These unlabeled examples are also added to the training set.
- The *test* set contains the unlabeled examples but the original labels are used for evaluation.

5.4 Conducting Experiments

5.4.1 Datasets

The performance of the multi-view teaching algorithm is evaluated on well-known datasets – "diabetes", "Iris", "Red Wine Quality", "contraceptives", "Yeast", which can be obtained from the UCI Machine Learning Repository [56]. The next section will describe the performance of the algorithm based on these datasets.

All datasets have real-valued attributes, there are no missing values. These datasets were preprocessed by the authors (cleansing, transformation, etc.). Detailed information about the datasets can be seen in Table 5.

- "Diabetes" is a dataset, which consists of 768 examples and 2 classes. The probability of randomly assigning a correct classification of an example is ½ (2 classes - 50% accuracy).
- The "*Iris"* dataset contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The probability of randomly assigning a correct classification of an example is 33%.
- "*Red Wine Quality"* models wine preferences based on physicochemical properties. It has 1600 examples, 11 attributes and 6

classes. The probability of randomly assigning a correct classification of an example is 17%.

- "*Contraceptives*" predicts the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics. It consists of 1473 instances, 3 classes and 9 real-valued attributes. The probability of randomly assigning a correct classification of an example is 33%.
- "*Yeast*" contains 1484 examples, divided in 10 classes. The number of attributes is 9. The probability of randomly assigning a correct classification of an example is 10%.

Table 5 Datasets

DataSet	*Classes*	*Attributes*	*Examples*	*Accuracy (random choice)*	*Accuracy (random) - based on the distribution*
Iris	3	4	150	33.33%	33.33%
Diabetes	2	8	768	50.00%	54.56%
Red Wine Quality (2009)	6	11	1600	16.67%	35.73%
contraceptives	3	9	1473	33.33%	35.38%
Yeast	10	8	1484	10.00%	22.32%

5.4.2 Experimental Results

The two algorithms are compared based on the percentage of labeled examples used for training. It was experimented with the following samples of labeled examples: 5%, 10%, 15%, 20%, 50%, 90%. Detailed results can be seen in Table 6.

The threshold of the algorithm, which was used was set to -16 (negative because of the logarithmic probability). It was experimented with various thresholds. When the threshold was too high, it made the classification accuracy of the training algorithm close to the supervised one, because few examples could surpass it. When the threshold was rather low, too many unlabeled examples managed to affect the performance of the weaker classifier, diminishing its classification accuracy. The value of this threshold was of greatest significance.

It can be seen from the table that the multi-view teaching algorithm outperforms the supervised learner.

Table 6 MVTA vs Naïve Bayes Classifier

Labeled examples	Algorithm	5%	10%	15%	20%	50%	90%
Iris	NB	80.07	89.81	93.37	94.03	95.08	94.82
Iris	**MVTA**	**82.56**	**90.77**	**93.53**	**94.10**	**95.08**	**94.82**
Yeast	NB	35.53	38.59	44.44	46.70	48.94	50.30
Yeast	**MVTA**	**36.16**	**42.15**	**45.55**	**47.78**	**48.98**	**50.68**
contraceptives	NB	69.74	71.97	72.96	73.53	75.55	80.04
contraceptives	**MVTA**	**70.00**	**72.19**	**73.16**	**73.71**	**75.70**	**82.76**
Red Wine Quality	NB	32.67	36.31	39.32	41.97	47.09	49.39
Red Wine Quality	**MVTA**	**32.40**	**36.63**	**40.33**	**43.34**	**52.16**	**52.16**
Diabetes	NB	69.74	71.97	72.96	73.53	75.55	80.04
Diabetes	**MVTA**	**70.00**	**72.19**	**73.16**	**73.71**	**75.70**	**82.76**

It can be seen from the table that the higher the percentage of labeled instances used for learning, the better the classification accuracy. On the other hand, the table shows that using fewer labeled examples can also bring relatively good results (Iris: 82.56% accuracy using only 5% labeled instances). The question is whether we can afford the process of labeling an example or not.

It can also be seen from the table that the semi-supervised multi-view teaching algorithm outperforms its supervised equivalent (the naïve Bayes classifier), but the improvement is at most 3%. When both classifiers are weak (Red

Wine Quality) the multi-view teaching algorithm fails to improve the classification accuracy (the two views are both weak). The reason that stands behind these results is that the two views are not conditionally independent given the classification function. Still, as Blum and Mitchell point out, semi-supervised multi-view learning can be effective.

Fig. 17 shows the comparison between the two algorithms based on the *Iris* dataset. The X axis corresponds to the percentage of labeled examples used in the training set. It can be seen that when the number of labeled examples is small (5%), the multi-view teaching algorithm outperforms its supervised equivalent with 2.49% (classification accuracy: 82.56%). Using 90% of the labeled examples brings classification accuracy of 94.82%, but both algorithms have the same results; the multi-view teaching algorithm does not impose an improvement in the classification accuracy. The rest of the examples (10% unlabeled examples) do not bring any new information, even though the values of their classification function are correctly predicted by the stronger learner. There is an upper bound that restricts the performance of the semi-supervised algorithm. The semi-supervised algorithm cannot outperform its supervised equivalent, which is based on 100% labeled examples.

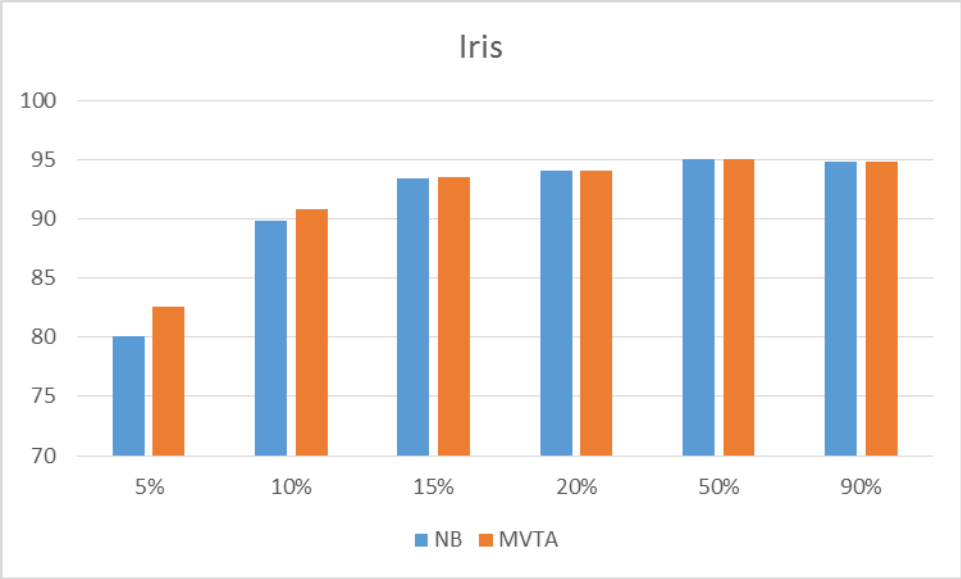


Fig. 17. Iris – classification accuracy, based on the percentage of labeled examples

It can be seen from Table 6 that the *yeast* dataset also follows this pattern: when the number of labeled examples is small, the improvement is the highest. Still, the rest of the datasets do not follow it. One reason is that even when using 90% of the labeled examples the classification accuracy (*Red Wine Quality*: 52.60%, *contraceptives*: 82.76%, *diabetes*: 82.76%) is not that high and the optimal one was not reached. There are representative unlabeled examples and their classifications are predicted most probably correctly by the multi-view teaching algorithm. Using 90% labeled examples brings better classifiers (compared to those learned using 5% labeled examples) and these classifiers have not reached their upper limits yet.

5.5 Conclusion

In this part a multi-view teaching algorithm was proposed. It uses two views and unlabeled examples. It is a modification of the standard co-training algorithm. It augments only the set of labeled examples of the weaker classifier.

The aim of the experiment was to "simulate" a semi-supervised framework by using only a fraction of the examples as labeled instances. Multiple well-known datasets were explored - "diabetes", "Iris", "Red Wine Quality", "contraceptives",

"Yeast". These datasets do not contain genuine views which are conditionally independent given the class. Still, the algorithm proved to be effective. In the semi-supervised scenario the performance of the proposed multi-view teaching algorithm was evaluated and compared to a supervised one (naïve Bayes classifier). The results prove that using multiple views and unlabeled examples improves the classification accuracy.

The promising results from the previous section inspired the application of the multi-view teaching algorithm to image segmentation (Part 6) – a teacher labels few pixels of an image beforehand and the rest of the pixels are used as unlabeled.

6. A Semi-supervised Image Segmentation System (SSISS)

6.1 Image Segmentation

Image segmentation [67] is the process of partitioning a digital image into regions or categories, which correspond to different classes. Every pixel in an image is allocated to one of the categories. Pixels with the same label (classification) share common visual characteristics.

In image segmentation various approaches exist:

- *Thresholding*

It is based on a threshold value and transforms a gray-scale image into a binary image. The important part in this method is to select the threshold value, which separates the classes.

- *Otsu's method* [17] (maximum variance).

The Otsu's method performs clustering-based image thresholding. It reduces a gray-scale image to a binary image. The algorithm assumes that the image contains two classes of pixels - foreground pixels and background pixels. It calculates the optimum threshold separating the two classes so that the intra-class variance is minimal or the inter-class variance is maximal. Otsu's method is a discrete analog of Fisher's Discriminant Analysis.

- *Different unsupervised clustering methods* (EM-algorithm [52], kMeans [53]).

Object segmentation [33, 68] extracts visual objects from an image. For example, in Fig. 18 the two images are manually segmented so that the two flowers are detected and extracted.

Fig. 18 Manual Object Segmentation

The difference between object detection (face detection, detection of dogs, cats, etc.) and object segmentation is that the former is interested in only the presence or absence of an object in an image. Usually, there is a frame marking the placement of the object, whereas in object segmentation the mask (the desired segmentation) is the important part.

Unsupervised object segmentation techniques decompose an image into the following classes: “object” and “background”, using features like: color, texture, shape, etc. The image is segmented in such a way that a mask can further be used differentiating between the segmented objects. There are no informed criteria for the type of objects we would like to extract. These image segmentation approaches do not rely on prior information. The algorithms rely only on the nature of the examples in order to segment the images.

If we performed a pure supervised segmentation, i.e. a classification algorithm, it is obvious we would need a training set, from which the classifier could learn. Because images are getting larger and larger and manual segmentation takes a lot of time, such a dataset is hard to collect. Imagine an expert labeling 1000 images and defining which pixel in each image belongs to the class “pizza” and which does not.

In such cases, when a labeled set of examples is difficult to collect, a semi-supervised approach can be used.

Semi-supervised image segmentation makes use of unlabeled examples. Furthermore, multiple views can be explored so that the classification accuracy is improved. Regarding the need for such segmentation, a learning system in the field of semi-supervised multi-view learning was developed and evaluated.

6.2 Semi-supervised image segmentation

Semi-supervised image segmentation concerns how to obtain the segmentation from a partially labeled image. A "teacher" should label few points of each class, giving the algorithm the idea of the clusters. The aim is to augment the training set with more labeled examples, reaching a better predictor. For instance, in Fig. 19 we want to learn the following two classes – "flower" and "not flower". The teacher has already labeled two red points (class "flower") and 2 black ones (class "not flower"). There is no supervised information concerning the rest of the pixels. Machine learning regards this problem as a binary classification problem – based on the labeled and unlabeled examples a function should be learned, which assigns 1/0 ("flower" and "not flower") to each unlabeled pixel.

Since we are using the learned function to obtain predictions only for the unlabeled examples in the image and there are no new examples, which are not in the image, it makes the machine learning problem a trunsductive one.

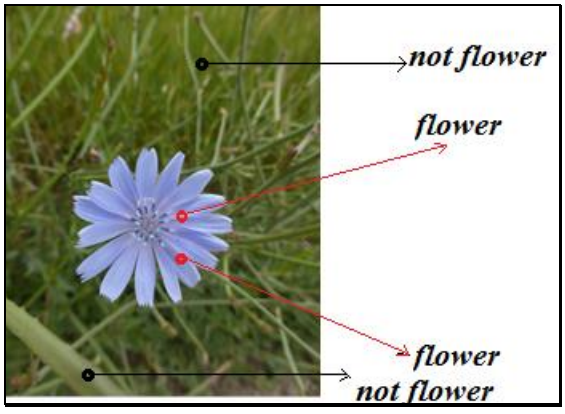


Fig. 19 Four labeled examples

There can be more than two classes and objects we would like to learn. It can be seen in Fig. 20 that the number of classes is three: “sky”, “building”, “tree”.

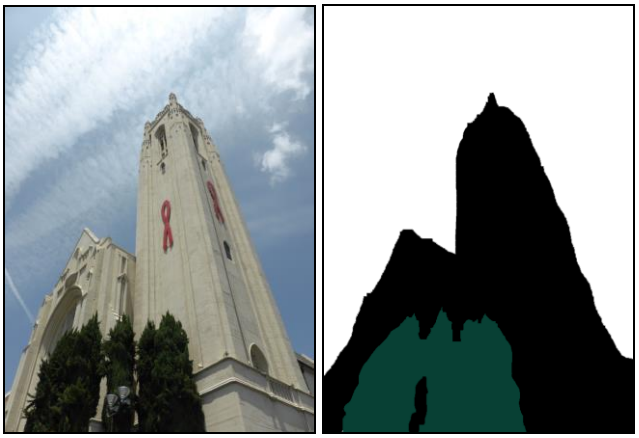


Fig. 20 Learning Three Objects

The “*teacher*” decides how many the number of classes will be - he is the one who defines beforehand what objects he wants to learn and segment. There might be various potential classes in an image.

6.3 Characteristics of SSISS

SSISS provides the user with semi-manual segmentation.

- SSISS uses multiple sources of information.
  - the first view contains the coordinates of the pixels (*x, y*):

    $$view_1 = (X, Y)$$

  - the second view contains the RGB values of the pixels (red, green, blue values ranging from 0 to 255:

    $$view_2 = (R, G, B)$$

The classifier, based on $view_2$ which consists of the RGB values, is especially susceptible to insufficient training examples. $view_1$ is the stronger one, as we usually want to segment objects, consisting of pixels, which are close to each other.

- SSISS benefits from unlabeled examples. A "*teacher*" labels only a small portion of the examples, the rest of the pixels are used as unlabeled instances (semi-manual segmentation).
- SSISS is based on the proposed in part 5 multi-view teaching algorithm

6.4 Software Architecture

The developed semi-supervised image segmentation system was designed and tested on a DELL, XPS laptop - 8GB RAM, Intel i7-2630QM processor under Windows 7. It was written in C++, MS Visual Studio 2012. The architecture of the system can be seen in Fig. 21. Two external libraries were used – openCV and GSL.

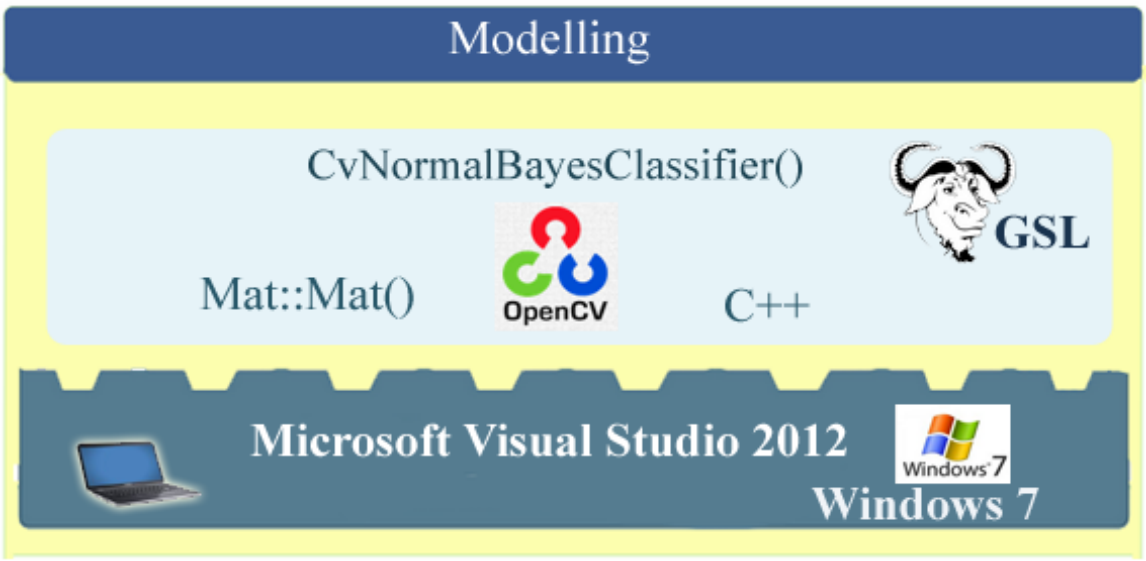


Fig. 21 Architecture of the image segmentation system

6.4.1 External Libraries

OpenCV [34] (Open Source Computer Vision) is an image processing library of programming functions for the computer vision society. Originally, it was developed by Intel research center in Nizhny Novgorod. It is also famous for its hardware optimization, especially for computers with Intel processors.

There is a machine learning module (MLL), which includes a set of classes and functions for statistical classification, regression, and clustering of data. The majority of the classification and regression implementations are C++ classes.

1. *Loading an image with OpenCv*

The class `Mat` can be used to store real or complex-valued vectors and matrices, grayscale or color images, and was used for pixel storing.

```
Mat src = imread("image1.jpg");
```

Some useful functions concerning the `Mat` class are:

- `Mat::col`(int **x**) : returns the column with index x
- `Mat::row`(int **x**) : returns the row with index x
- `Mat::operator()`(Range **rowRange**, Range **colRange**) : returns a sub-matrix and is a generalized form of Mat::row() and Mat::col()

For example:

```
Mat view1 = train(Range(0, l), Range(0, VIEW_1));
```

all labeled examples (*l* is the number of labeled examples) from the train set *train* are returned but only the first `VIEW_1` number of attributes are included in the submatrix.

- `Mat::zeros` (int **rows**, int **cols**, int **type**) : creates a matrix with zeros
- `Mat::resize`(size_t **sz**) : changes the number of rows
- `Mat::push_back`(const Mat& **m**) : adds elements to the bottom of the matrix, the number of columns must be the same as in the container matrix.
- `Mat::pop_back`(size_t **nelems**=1) : removes rows from the bottom of the matrix

2. *Naïve Bayes Classifier*

The built-in implementation of the naïve Bayes classifier was used in the image segmentation system - it estimates the most probable classifications of the unlabeled examples.

```
CvNormalBayesClassifier classifier;
classifier.train(lab, classes, Mat(), Mat(), false);
y = classifier.predict(unlabelled);
```

GSL (GNU Scientific Library) is a numerical library for C++ developers and it is a free software under the GNU General Public License. The library provides a wide variety of mathematical operators (more than 1000 functions). In the system the random number generator was used.

- `unsigned long int` **`gsl_rng_uniform_int`** `(const gsl_rng * r, unsigned long int n)`

This function returns a random integer varying in the range [0, n-1]. All values in the range [0, n-1] are produced with equal probability.

6.4.2 Implemented Modules

The following modules were implemented, as it can be seen in Fig. 22:

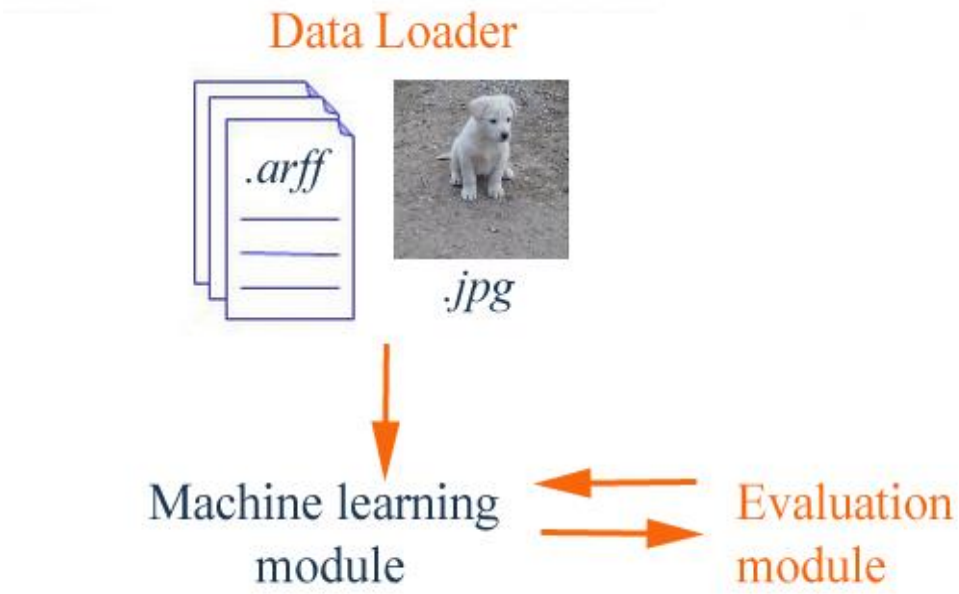


Fig. 22 Semi-supervised Segmentation - Modelling

1. *Data loader* module:

```
template<class ATTR_TYPE>
void MVT<ATTR_TYPE>::loadData(char* f)
```

   - Attribute-relation File Format (.arff) parser for loading arff files.
   - Image loader – using the functionality of opencv

2. *Machine learning* module

- Naïve Bayes classifier – using the functionality of opencv
- Bayes classifier, based on Multivariate normal distribution

```
template<class ATTR_TYPE>
void MVT<ATTR_TYPE>::learnSupervised()
```

- Multi-view teaching algorithm

```
template<class ATTR_TYPE>
void MVT<ATTR_TYPE>::MVTCombine()
```

3. Evaluation Module

Semi-supervised evaluation

```
template<class ATTR_TYPE>
double MVT<ATTR_TYPE>::testMVT(LEARNER<ATTR_TYPE>& L1 ,
                               LEARNER <ATTR_TYPE>& L2)
```

Supervised evaluation

```
template<class ATTR_TYPE>
double MVT<ATTR_TYPE>::testNB(LEARNER<ATTR_TYPE>& L)
```

6.5 Evaluation of SSISS

6.5.1 Dataset

It was experimented with three images – Fig. 23, Fig. 24 and Fig. 25.

The original images were segmented by a teacher into pre-defined classes. In Fig. 23, 24 and 25 (a) is a sample image and (b) - the distribution of pixels between the classes. Image (b) was used for evaluation of the multi-view teaching algorithm and represents the desired classifications of image (a)'s pixels.

Fig. 23 (a) - original image, (b) – desired segmentation

Fig. 24 (a) - original image, (b) – desired segmentation

Fig. 25 (a) – original image, (b) – desired segmentation

6.5.2 Training and test sets

As in the previous section the training set consists of a fraction of labeled examples from the original dataset. A small amount of pixels is chosen randomly; these pixels are added to D_1. The rest of the instances have their classifications removed. These unlabeled examples are added to D_2. A final training set is constructed:

$$D = D_1 \cup D_2.$$

The test set contains all the examples in D_2. We will evaluate the multi-view teaching algorithm based on these examples, by using the original classifications with respect to image (b).

6.5.3 Monte Carlo Cross-validation

For classification error estimation a Monte Carlo cross-validation was used, multiple random samples were averaged. A k-folded cross-validation can also be applied. As we require a limited and pre-defined size of labeled examples, the parameter k depends on it. The examples should be divided into folds, so that one fold contains all the labeled examples. K is the number of all examples divided by the number of labeled ones. The smaller the number of labeled examples, the bigger the number of folds. Furthermore, to reduce variability, multiple rounds of cross-

validation steps should be performed using different partitions, and the validation results should be averaged over the rounds.

6.5.4 Experimental Results

- *Comparison of the multi-view teaching algorithm, based on naïve Bayes classifiers (for the underlying learners) to a supervised naïve Bayes classifier:*

The image from Fig. 23 consists of 50700 pixels. At each cross-validation step only a small amount of labeled pixels is used. Multiple tests were held depending on the number of labeled examples (4, 6, 10, 16, 20, 50 pixels). For example, the teacher labels four points in the image (two for each class) and the rest 50696 examples are used as unlabeled.

The performance of the multi-view teaching algorithm (MTA) based on Naïve Bayes Classifiers is compared to its supervised equivalent (NB), depending on the number of labeled examples. The average classification accuracy after the process of cross-validation can be seen in Table 7.

Table 7 MTA vs NB, based on the number of labeled pixels

Algorithm	4	6	10	16	20	50
NB	63.30%	76.23%	85.44%	89.57%	90.33%	92.37%
MTA	**68.62%**	**81.30%**	**88.14%**	**90.74%**	**91.24%**	**92.51%**

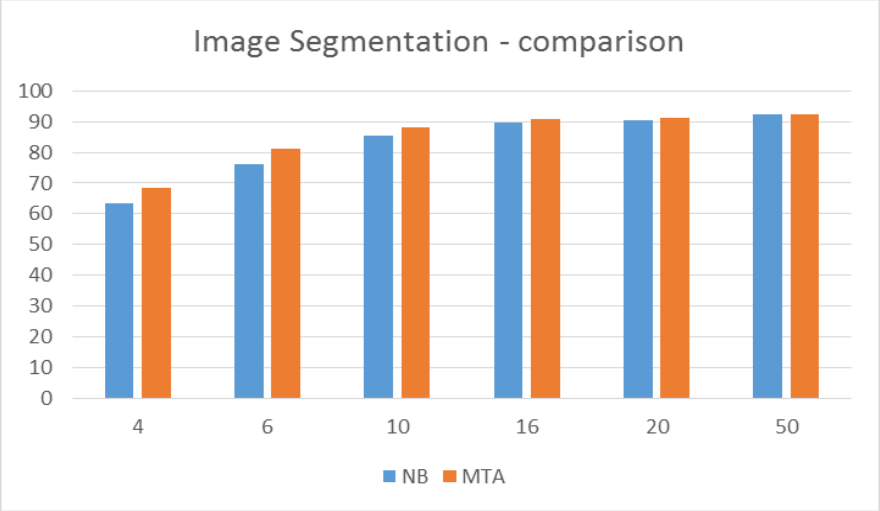


Fig. 26 MTA vs NB, based on the amount of labeled examples

It can be seen from Fig.26 that the multi-view teaching algorithm outperforms the Naïve Bayes classifier. It dominates especially when the number of labeled examples is small and the supervised classifier suffers from insufficient number of training examples. It can also be seen in Fig. 26 that using 50 labeled examples (out of 50700 pixels) brings close results (for both algorithms) (MTA:92.51% vs NB:92.37%), which was already discussed in part 5.

The performance of the algorithms for the three images (Fig. 23, Fig. 24, and Fig. 25) is compared in Table 8. Sixteen initial points are labeled (8 for each class). It was experimented with multiple thresholds and the one chosen was such that at least 10 examples could surpass it (used as labeled examples).

Table 8 MTA vs NB, 16 labeled examples

	MTA	NB
Image 1	**90.74%**	89.57%
Image 2	**80.76%**	78.82%
Image 3	**90.10%**	89.12%

The cross-validation steps would be hard to perform, without the optimization of the openCV library. Little research has been performed in this area so far, due to the tedious evaluation process, the small amount of labeled examples and the large number of cross-validation steps.

The performance of the algorithm is evaluated based on different samples of randomly chosen labeled examples. The teacher, who will be responsible for this process is expected to label representative examples which will lead to better classification accuracy.

Another point worth mentioning, is execution speed. It has already been mentioned that the original co-training algorithm is not appropriate for datasets consisting of large amounts of examples. For example, an 18-megapixels image has 4896x3672 points, which makes 17,978,112 examples in the dataset. Another major execution speed advantage of the multi-view algorithm is the partition of attributes into views which can also impose a major speedup for some classification algorithms (the complexity depends on the number of attributes used for learning). 'Divide and conquer' has been a positive strategy in many informatics fields and it is a cure for the curse of dimensionality.

- *Comparison of the multi-view teaching algorithm, based on multivariate normal distribution (MND-MVTA) and a Bayesian supervised classifier based on multivariate normal distribution (MND-SL):*

The performance of the multi-view teaching algorithm based on two *MND-SL* classifiers has also been evaluated for the images from Fig. 23, Fig. 24, and Fig. 25. Sixteen initial labeled examples were used. It was experimented with multiple thresholds and the one chosen could be surpassed by at least 10 examples, which could then be used as labeled examples. The performance of the algorithms is compared in Table 9. It can be seen from the table that the multi-view teaching algorithm outperforms its supervised equivalent. The reason for the scope of the improvements lies in the very multivariate distribution, which is also heavily influenced by the small (insufficient) number of labeled examples.

A fine-tuning procedure was also used so that the covariance matrix becomes invertible (*eps* is a small number):

$$\sum_y = \sum_y + eps$$

Table 9 Comparison of the two algorithms, 16 initial labeled examples

Algorithm	*MND-MVTA*	*MND-SL*
Image 1	**84.36%**	79.22%
Image 2	**79.14%**	73.74%
Image 3	**86.02%**	80.18%

6.6 Conclusion

In semi-supervised image segmentation there is an expert in the field who is responsible for labeling few pixels of an image. For example, a Photoshop user selects a small amount of pixels with the mouse and defines what classes they belong to. The rest of the pixels are used as unlabeled.

The proposed in part 5 semi-supervised multi-view teaching algorithm was applied in the system and compared to a supervised equivalent. Two types of classifiers were used for the underlying learners: a naive Bayes classifier and a Bayes classifier, based on multivariate normal distribution. The results show that the semi-supervised algorithm outperforms its supervised equivalent and is especially effective when the number of labeled examples is small.

Conclusion

A predictive system is an autonomous agent that relies on past history, or a training set of labeled examples and learns a classifier in order to forecast events that will happen in the future (simulates intelligent behavior). Yet, there are areas of research, where there is not enough labeled data to train a good classifier. The current dissertation focuses on such problems and uses multiple sources of information in conjunction with unlabeled examples.

Two approaches to semi-supervised multi-view learning are proposed (*the semi-supervised multi-view genetic algorithm* and *the multi-view teaching algorithm*). They are further applied to two domains – sentiment analysis in Bulgarian and semi-supervised image segmentation. The importance of these fields inspired the implementation of two learning system:

- SASBG - a semi-supervised multi-view learning system for sentiment analysis from texts in the Bulgarian language;
- SSISS - a semi-supervised multi-view learning system for image segmentation.

The Bulgarian language and sentiment analysis applied to it are an area that has been little researched, and insufficient information has been a burning problem. In this area, there are not enough labeled examples. To overcome these difficulties, unlabeled examples are used and the proposed *semi-supervised multi-view genetic algorithm* is applied. The *World Wide Web* abounds in English labeled examples; they are easy to find. The labeled instances were automatically translated into Bulgarian with Google translate so that a second view is formed. SASBG also uses genuine Bulgarian unlabeled instances. These unlabeled reviews were translated into English. The results from the previous section show that using unlabeled examples decreases the error of the system. The semi-supervised multi-view algorithm outperforms its supervised competitors.

Since genetic algorithms are iterative procedures, taking into consideration the large number of features and number of examples, in SASBG a modern distributed framework was developed. This system makes use of technologies like Hadoop and Spark, which played a major role in the evaluation of the performance of the algorithm.

SSISS uses the proposed in this dissertation *multi-view teaching algorithm*. It requires prior knowledge about the classes, an expert who labels a few examples beforehand. The rest of the pixels are used as unlabeled. Two views are used. The first view contains the coordinates of the pixels and the second – the RGB values of the pixels. The results from the previous section show that using unlabeled examples in conjunction with multiple sources of information improves the classification accuracy of the algorithm. The multi-view teaching algorithm outperforms its supervised competitor.

Future Works

1. SASBG
    - It should be experimented with more application domains, not only movie reviews. Another hot area in sentiment analysis is sentiment analysis of hotel/location/holiday reviews in Bulgarian.
    - It should be experimented with labeled and unlabeled examples from heterogeneous domains. Domain propagation should be explored, so that we can train the system in one domain and apply it to a different one.
    - More training and test examples should be collected and used as the system is able to support a large amount of data.
    - More translation tools should be explored – not only Google translate, but Reverso, Linguee, etc. They can form even more views; currently it is experimented with only two views.

2. SSISS:

- The system should be tested with more images – even thousands. Such a domain of classified (segmented images) is hard to find.
- It should be further ported for parallel computation. Considering the large size of images nowadays, algorithms running on a cluster of machines are required. Space cameras can record multiple images in a second with high resolution.

Scientific Contributions

- Two new methods for semi-supervised multi-view learning are developed:
  - *the semi-supervised multi-view genetic algorithm*.
    The common error, which depends on multiple sources of information is minimized.
  - *the multi-view teaching algorithm*. It is a modification of the standard co-training algorithm and improves only the weaker of the classifiers.
- Design of experiments - so that the proposed methods can be evaluated;
- Conducting experiments - tailor-made datasets were collected for the purposes of the experiments;
- Analysis of the experimental results.

Applied Contributions

- SASBG - a semi-supervised multi-view learning system for sentiment analysis in the Bulgarian language is designed and developed. It is a modern distributed system and parallel calculations can be executed simultaneously;
- SSISS - a semi-supervised multi-view image segmentation system is designed and developed;

Publications

[1] Lazarova, Gergana, and Ivan Koychev. "Semi-supervised Multi-view Sentiment Analysis." *Computational Collective Intelligence*. **Springer International Publishing**, **2015**. 181-190.

[2] Gergana Lazarova and Todor Tsonkov. Sentiment analysis of Texts in Bulgarian. **Journal of the Technical University** – Sofia, *TECHSYS* 2015, BULGARIA, Plovdiv. 203-207.

[3] Lazarova, Gergana, and Ivan Koychev. "A Semi-Supervised Multi-view Genetic Algorithm." *Artificial Intelligence, Modelling and Simulation (AIMS), 2014 2nd International Conference on*. **IEEE, 2014**. 87 - 91

[4] Lazarova, Gergana Angelova. "Semi-supervised Image Segmentation."*Artificial Intelligence: Methodology, Systems, and Applications*. **Springer International Publishing, 2014**. 59-68.

[5] Gergana Lazarova, Milen Chechev, and Ivan Koychev. "A Multi-View Learning Algorithm". *Education and Research in the Information Society*, Plovdiv, May, 2014, 131-139

[6] Gergana Lazarova and Ivan Koychev. "Parallel Semi-supervised EM-algorithm (MT-SSEM)". *National Conference on Education in the Information Society*, Plovdiv, May, 2013, 149-157

Workshops and short papers

[1] Gergana Lazarova. "Emergent learning system of multiple views", *Lab Surfing workshop*, Thessaloniki, March 2014.

[2] Gergana Lazarova and Ivan Koychev. "Sentiment analysis using multiple sources of information". *Big Data in eLearning and Digital Collections International Workshop*, 29 June 2015, IICT-BAS Acad. G. Bonchev Str. Block 25A.

[3] Гергана Лазарова, Иван Койчев. "Генетичен алгоритъм за учене от много източници на данни". *125 години математика и природни науки в Софийски университет "Св. Климент Охридски"*, София, 2014. 43-45

Citations

Paper [5] is cited 1 time in:

[1] S. Li, R. Ju, T. Ren, and G. Wu. "Saliency cuts based on adaptive triple thresholding". *International Conference on Image Processing* (ICIP), IEEE Signal Processing Society, 2015.

Participation in Projects

[1] Smart Book: 2009 – 2012. The aims of the project is to develop a range of intelligent functions supporting both the e-books author and the reader. The project is funded by the Bulgarian Ministry of Education (Fund “Science”).
http://dse.fmi.uni-sofia.bg/SmartBook/team.htm

[2] "Формиране на нова генерация от изследователи в областта на математиката, информатиката и компютърните науки чрез подкрепа на творческия и иновативен потенциал на докторанти, постодокторанти и млади учени.”, Sofia University “St. Kliment Ohridski”, Faculty of mathematics and informatics. This project is supported by the European Social Fund through the Human Resource Development Operational Programme under contract BG051PO001-3.3.06-0052 (2012/2014).
http://sci.fmi.uni-sofia.bg/node/41

Declaration

With regard to the previous sections, the author declares that in reference to conferring of academic and scientific degree DOCTOR in professional field 4.6 "Informatics and Computer Science", at Sofia University "St. Kliment Ohridski":

- the dissertation "Learning from multiple sources of data" is an original work except where specifically indicated in the text and bibliography. The use of previous results is fairly rejected with appropriate references and complies with the copyrights of their authors, publishers or copyright holders.
- The results and contributions of the thesis (scientific and applied) are original and are not taken from other works the author has not participated in.
- The dissertation is not submitted for a degree, diploma or other qualification at another university or research institute.

References

[1] Searle, John R. *Mind, language and society: Philosophy in the real world*. New York: Basic books, 1998.

[2] Olivier Chapelle, Bernhard Schölkopf and Alexander Zien. *Semi-supervised Learning*, MIT Press, 2006

[3] Blum, Avrim, and Tom Mitchell. "Combining labeled and unlabeled data with co-training." *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 1998, 92-100

[4] Nigam, Kamal, and Rayid Ghani. "Analyzing the effectiveness and applicability of co-training." *Proceedings of the ninth international conference on Information and knowledge management*. ACM, 2000, 86-93.

[5] Belkin, Mikhail, Partha Niyogi, and Vikas Sindhwani. "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples." *The Journal of Machine Learning Research* 7 (2006): 2399-2434.

[6] Wan, Xiaojun. "Bilingual co-training for sentiment classification of Chinese product reviews." *Computational Linguistics* 37.3 (2011): 587-616.

[7] Levin, Anat, Paul Viola, and Yoav Freund. "Unsupervised improvement of visual detectors using cotraining." *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003. 626-633.

[8] Tang, Feng, et al. "Co-tracking using semi-supervised support vector machines." *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007. 1-8.

[9] Cheng, Jian, and Kongqiao Wang. "Active learning for image retrieval with Co-SVM." *Pattern recognition* 40.1 (2007): 330-334.

[10] Gupta, Sonal, et al. "Watch, listen & learn: Co-training on captioned images and videos." *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2008. 457-472.

[11] Brefeld, Ulf, and Tobias Scheffer. "Co-EM support vector learning."*Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004. 16.

[12] Dasgupta, Sanjoy, Michael L. Littman, and David McAllester. "PAC generalization bounds for co-training." *Advances in neural information processing systems* 1 (2002): 375-382.

[13] Xu, Chang, Dacheng Tao, and Chao Xu. "A survey on multi-view learning."*arXiv preprint arXiv:1304.5634* (2013).

[14] Mihalcea, Rada. "Co-training and self-training for word sense disambiguation."*Proceedings* Introduction to semi-supervised learning *of the Conference on Computational Natural Language Learning (CoNLL-2004)*. 2004.

[15] Zhu, Xiaojin, and Andrew B. Goldberg. "Introduction to semi-supervised learning." *Synthesis lectures on artificial intelligence and machine learning* 3.1 (2009): 1-130.

[16] Nigam, Kamal, Andrew McCallum, and Tom Mitchell. "Semi-supervised text classification using EM." *Semi-Supervised Learning* (2006): 33-56.

[17] Otsu, Nobuyuki. "A threshold selection method from gray-level histograms."*Automatica* 11.285-296 (1975): 23-27.

[18] Sahami, Mehran, et al. "A Bayesian approach to filtering junk e-mail." *Learning for Text Categorization: Papers from the 1998 workshop*. Vol. 62. 1998. 98-105.

[19] Abdi, Hervé, and Lynne J. Williams. "Principal component analysis." *Wiley Interdisciplinary Reviews: Computational Statistics* 2.4 (2010): 433-459.

[20] Witten, Ian H., and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[21] Liu, Bing. "Sentiment analysis and subjectivity." *Handbook of natural language processing* 2 (2010): 627-666.

[22] Kennedy, Alistair, and Diana Inkpen. "Sentiment classification of movie reviews using contextual valence shifters." *Computational intelligence* 22.2 (2006): 110-125.

[23] Taboada, Maite, et al. "Lexicon-based methods for sentiment analysis."*Computational linguistics* 37.2 (2011): 267-307.

[24] Darwin, Charles. "On the origins of species by means of natural selection."*London: Murray* (1859): 247.

[25] Forbes, Nancy. *Imitation of life: how biology is inspiring computing*. Cambridge: Mit Press, 2004.

[26] Mitchell, Melanie. *An introduction to genetic algorithms*. MIT press, 1998.

[27] Tikhonov, Andrey. "Solution of incorrectly formulated problems and the regularization method." *Soviet Math. Dokl.*. Vol. 5. 1963. 1035-1038.

[28] Salton, Gerard, Anita Wong, and Chung-Shu Yang. "A vector space model for automatic indexing." *Communications of the ACM* 18.11 (1975): 613-620.

[29] Cunningham, Hamish, Diana Maynard, and Kalina Bontcheva. *Text processing with gate*. Gateway Press CA, 2011.

[30] McAuley, Julian, and Jure Leskovec. "Hidden factors and hidden topics: understanding rating dimensions with review text." *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013. 165-172.

[31] Holmes, Alex. *Hadoop in practice*. Manning Publications Co., 2012.

[32] Pentreath, Nick. "Machine Learning with Spark." (2014).

[33] Park, Soo Beom, Jae Won Lee, and Sang Kyoon Kim. "Content-based image classification using a neural network." *Pattern Recognition Letters* 25.3 (2004): 287-300.

[34] Suarez, Oscar Deniz, et al. *OpenCV Essentials*. Packt Publishing Ltd, 2014.

[35] Brefeld, Ulf, et al. "Efficient co-regularised least squares regression."*Proceedings of the 23rd international conference on Machine learning*. ACM, 2006. 137-144

[36] Hoerl, Arthur E., and Robert W. Kennard. "Ridge regression: Biased estimation for nonorthogonal problems." *Technometrics* 12.1 (1970): 55-67.

[37] Schmidt, Mark, Glenn Fung, and Rmer Rosales. "Fast optimization methods for l1 regularization: A comparative study and two new approaches." *Machine Learning: ECML 2007*. Springer Berlin Heidelberg, 2007. 286-297.

[38] Blum, Avrim, et al. "Semi-supervised learning using randomized mincuts."*Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004. 13.

[39] Sindhwani, Vikas, Partha Niyogi, and Mikhail Belkin. "A co-regularization approach to semi-supervised learning with multiple views." *Proceedings of ICML workshop on learning with multiple views*. 2005. 74-79

[40] Yu, Shipeng, et al. "Bayesian co-training." *The Journal of Machine Learning Research* 12 (2011): 2649-2680.

[41] Bickel, Steffen, and Tobias Scheffer. "Multi-View Clustering." *ICDM*. Vol. 4. 2004. 19-26.

[42] Wang, Wei, and Zhi-Hua Zhou. "Analyzing co-training style algorithms."*Machine Learning: ECML 2007*. Springer Berlin Heidelberg, 2007. 454-465.

[43] Argyriou, Andreas, Charles A. Micchelli, and Massimiliano Pontil. "When is there a representer theorem? Vector versus matrix regularizers." *The Journal of Machine Learning Research* 10 (2009): 2507-2529.

[44] Zhu, Xiaojin, Zoubin Ghahramani, and John Lafferty. "Semi-supervised learning using gaussian fields and harmonic functions." *ICML*. Vol. 3. 2003. 912-919.

[45] Breheny, Patrick, and Jian Huang. "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection." *The annals of applied statistics* 5.1 (2011): 232-253.

[46] Sarkar, Anoop. "Applying co-training methods to statistical parsing."*Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*. Association for Computational Linguistics, 2001. 1-8.

[47] Gong, Pinghua, et al. "A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems." *Machine learning: proceedings of the International Conference. International Conference on Machine Learning*. Vol. 28. No. 2. NIH Public Access, 2013. 37.

[48] Nigam, Kamal, et al. "Text classification from labeled and unlabeled documents using EM." *Machine learning* 39.2-3 (2000): 103-134.

[49] Golberg, David E. "Genetic algorithms in search, optimization, and machine learning." *Addion wesley* 1989 (1989).

[50] Banea, Carmen, Janyce M. Wiebe, and Rada Mihalcea. "A bootstrapping method for building subjectivity lexicons for languages with scarce resources." (2008).

[51] Whiteley, D. "Applying genetic algorithms to neural network problems." *Neural Networks* 1 (1988): 230.

[52] Carson, Chad, et al. "Blobworld: Image segmentation using expectation-maximization and its application to image querying." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.8 (2002): 1026-1038.

[53] Dehariya, Vinod Kumar, Shailendra Kumar Shrivastava, and R. C. Jain. "Clustering of image data set using k-means and fuzzy k-means algorithms."*Computational Intelligence and Communication Networks (CICN), 2010 International Conference on*. IEEE, 2010. 386-391.

[54] Christoudias, C., Raquel Urtasun, and Trevor Darrell. "Multi-view learning in the presence of view disagreement." *arXiv preprint arXiv:1206.3242* (2012).

[55] Christoudias, C. Mario, et al. "Co-training with noisy perceptual observations."*Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009. 2844-2851.

[56] Bache, Kevin, and Moshe Lichman. "UCI machine learning repository." (2013).

[57] Xu, Qing-Song, and Yi-Zeng Liang. "Monte Carlo cross validation."*Chemometrics and Intelligent Laboratory Systems* 56.1 (2001): 1-11.

[58] Tikhonov, Andrey Nikolayevich. "On the stability of inverse problems." *Dokl. Akad. Nauk SSSR*. Vol. 39. No. 5. 1943. 195-198.

[59] Li, Haibo, Yutaka Matsuo, and Mitsuru Ishizuka. "Graph based multi-view learning for CDL relation classification." *Semantic Computing, 2009. ICSC'09. IEEE International Conference on*. IEEE, 2009. 473-480.

[60] Li, Haibo, Yutaka Matsuo, and Mitsuru Ishizuka. "Semantic Relation Extraction Based on Semi-supervised Learning." *Information Retrieval Technology*. Springer Berlin Heidelberg, 2010. 270-279.

[61] Zhu, Xiaojin, and Zoubin Ghahramani. *Learning from labeled and unlabeled data with label propagation*. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002. 107.

[62] He, Jingrui, and Rick Lawrence. "A graph-based framework for multi-task multi-view learning." *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011. 25-32.

[63] Farquhar, Jason, et al. "Two view learning: SVM-2K, theory and practice."*Advances in neural information processing systems*. 2005. 355-362.

[64] Melacci, Stefano, and Mikhail Belkin. "Laplacian support vector machines trained in the primal." *The Journal of Machine Learning Research* 12 (2011): 1149-1184.

[65] Sun, Shiliang. "Multi-view Laplacian support vector machines." *Advanced Data Mining and Applications* (2011): 209-222.

[66] Г. Агре, З. Марков, Д. Дочев. Увод в машинното самообучение. Софтех, София 2001

[67] Pal, Nikhil R., and Sankar K. Pal. "A review on image segmentation techniques." *Pattern recognition* 26.9 (1993): 1277-1294.

[68] Pal, N. R., and Sankar K. Pal. "Object-background segmentation using new definitions of entropy." *Computers and Digital Techniques, IEE Proceedings E*136.4 (1989): 284-295.

[69] Kiritchenko, Svetlana, and Stan Matwin. "Email classification with co-training."*Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*. IBM Corp., 2011. 301-312.

[70] Lappalainen, Harri, and J. Miskin. "Ensemble learning, Advances in Independent Component Analysis." (2000): 75-92.

[71] Matsubara, Edson Takashi, Maria Carolina Monard, and Gustavo EAPA Batista. "Multi-view Semi-supervised Learning: An Approach to Obtain Different Views from Text Datasets." *LAPTEC*. 2005. 97-104.

[72] Sindhwani, Vikas, and David S. Rosenberg. "An RKHS for multi-view learning and manifold co-regularization." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008. 976-983.

[73] Sun, Shiliang, Feng Jin, and Wenting Tu. "View construction for multi-view semi-supervised learning." *Advances in Neural Networks–ISNN 2011*. Springer Berlin Heidelberg, 2011. 595-601.

[74] Wan, Xiaojun. "Co-training for cross-lingual sentiment classification."*Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, 2009. 235-243.

[75] Wang, Wei, and Zhi-Hua Zhou. "A new analysis of co-training." *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010. 1135-1142.

[76] Wang, Zhe, Songcan Chen, and Daqi Gao. "A novel multi-view learning developed from single-view patterns." *Pattern Recognition* 44.10 (2011): 2395-2413.

[77] Zhou, Zhi-Hua, and Ming Li. "Semi-Supervised Regression with Co-Training."*IJCAI*. Vol. 5. 2005. 908-913.

[78] Zhou, Zhi-Hua, De-Chuan Zhan, and Qiang Yang. "Semi-supervised learning with very few labeled training examples." *Proceedings of the national conference on artificial intelligence*. Vol. 22. No. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007. 675.

[79] Goldberg, Andrew B., and Xiaojin Zhu. "Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization."*Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*. Association for Computational Linguistics, 2006. 45-52.

[80] Liu, Bing. "Sentiment Analysis and Opinion Mining Morgan & Claypool Publishers." (2012).

[81] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques." *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002. 79-86.

[82] Sindhwani, Vikas, and Prem Melville. "Document-word co-regularization for semi-supervised sentiment analysis." *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008. 1025-1030.

[83] Hu, Minqing, and Bing Liu. "Mining and summarizing customer reviews."*Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004. 168-177.

[84] Culp, Mark, and George Michailidis. "Graph-based semisupervised learning."*Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30.1 (2008): 174-179.

[85] Bennett, Kristin, and Ayhan Demiriz. "Semi-supervised support vector machines." *Advances in Neural Information processing systems* (1999): 368-374.

[86] Fung, Glenn, and Olvi L. Mangasarian. "Semi-superyised support vector machines for unlabeled data classification." *Optimization methods and software*15.1 (2001): 29-44.

[87] Sun, Shiliang. "Semantic features for multi-view semi-supervised and active learning of text classification." *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*. IEEE, 2008. 731-735.

[88] Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the royal statistical society. Series B (methodological)* (1977): 1-38.

[89] Rosenberg, Chuck, Martial Hebert, and Henry Schneiderman. "Semi-supervised self-training of object detection models." (2005).

[90] Belkin, Mikhail, Partha Niyogi, and Vikas Sindhwani. "On manifold regularization." *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*. 2005. 17-24.

[91] Taboada, Maite, et al. "Lexicon-based methods for sentiment analysis."*Computational linguistics* 37.2 (2011): 267-307.

[92] Mitchell, Tom M. "Machine learning. WCB." (1997).

[93] Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." *CS224N Project Report, Stanford* 1 (2009): 12.

[94] Speriosu, Michael, et al. "Twitter polarity classification with label propagation over lexical links and the follower graph." *Proceedings of the First workshop on Unsupervised Learning in NLP*. Association for Computational Linguistics, 2011. 53-63.

[95] Reyes, Antonio, Paolo Rosso, and Tony Veale. "A multidimensional approach for detecting irony in twitter." *Language Resources and Evaluation* 47.1 (2013): 239-268.

[96] Blum, Avrim, and Shuchi Chawla. "Learning from labeled and unlabeled data using graph mincuts." (2001): 19.

[97] Zhou, Zhi-Hua, and Ming Li. "Semi-Supervised Regression with Co-Training."*IJCAI*. Vol. 5. 2005. 908-913.

[98] Nigam, Kamal, and Rayid Ghani. "Understanding the behavior of co-training."*Proceedings of KDD-2000 workshop on text mining*. 2000. 15-17.

[99] Chen, Minmin, Kilian Q. Weinberger, and John Blitzer. "Co-training for domain adaptation." *Advances in neural information processing systems*. 2011. 2456-2464.

[100] Chapelle, Olivier, Vikas Sindhwani, and Sathiya S. Keerthi. "Optimization techniques for semi-supervised support vector machines." *The Journal of Machine Learning Research* 9 (2008): 203-233.

[101] Chapelle, Olivier, et al. "Choosing multiple parameters for support vector machines." *Machine learning* 46.1-3 (2002): 131-159.

List of Figures

List of Tables