FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Machine Learning With Dual Process Models

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Martin Unger BSc
Matrikelnummer 0726109

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Ao. Univ.-Prof. Mag. Dr. Horst Eidenberger

Wien, 4. März 2015

Martin Unger	Horst Eidenberger

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Machine Learning With Dual Process Models

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Martin Unger BSc
Registration Number 0726109

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao. Univ.-Prof. Mag. Dr. Horst Eidenberger

Vienna, 4th March, 2015

Martin Unger

Horst Eidenberger

Erklärung zur Verfassung der Arbeit

Martin Unger BSc
Alliiertenstraße 5/17, 1020 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. März 2015

Martin Unger

Kurzfassung

Ähnlichkeitsmessung ist ein wichtiger Bestandteil der meisten Maschinenlern-Algorithmen. Traditionelle Ansätze richten ihr Hauptaugenmerk entweder auf taxonomisches oder thematisches Denken. Psychologische Forschung deutet aber darauf hin, dass eine Kombination beider Ansätze nötig ist, um eine menschenähnliche Ähnlichkeitsmessung zu erreichen. Diese Kombination nennt man Similarity Dual Process Model (DPM).

Diese Arbeit beschreibt, wie ein DPM als lineare Kombination von Distanz- und Ähnlichkeitsmaßen erzeugt werden kann. Wir nutzen Generalisierungsfunktionen, um Distanz in Ähnlichkeit umzuwandeln. DPMs sind Kernelfunktionen ähnlich. Deshalb können sie in jeden Maschinenlern-Algorithmus, der Kernelfunktionen nutzt, integriert werden. Um die Verwendung von DPMs zu fördern, stellen wir Implementierungen von Kernelfunktionen zur Verfügung.

Natürlich funktionieren nicht alle DPMs, die wir formulieren können, gleich gut. Deshalb testen wir die Leistung mit einer praktischen Anwendung: der Erkennung von Fußgängern in Bildern. Wir nehmen an, dass DPMs nur sinnvoll sind, sofern ihre Leistung als Ganzes besser ist als die ihrer Teile. In den Experimenten haben wir DPM-Kernel gefunden, die für die Testdaten eine mit konventionellen Kerneln vergleichbare Leistung erbracht haben. Wir stellen daher einen Baukasten zum Formulieren solcher Kernel bereit, um weitere Experimente in anderen Anwendungsbereichen des Maschinenlernens zu unterstützen.

Abstract

Similarity measurement processes are a core part of most machine learning algorithms. Traditional approaches focus on either taxonomic or thematic thinking. Psychological research suggests that a combination of both is needed to model human-like similarity perception adequately. Such a combination is called a Similarity Dual Process Model (DPM).

This thesis describes how to construct DPMs as a linear combination of existing measures of similarity and distance. We use generalization functions to convert distance into similarity. DPMs are similar to kernel functions. Thus, they can be integrated into any machine learning algorithm that uses kernel functions. To foster the use of DPMs, we provide kernel function implementations.

Clearly, not all DPMs that can be formulated work equally well. Therefore, we test classification performance in a real-world task: the detection of pedestrians in images. We assume that DPMs are only viable if they are better classifiers than their constituting parts. In our experiments, we found DPM kernels that matched the performance of conventional kernels for our data set. Eventually, we provide a construction kit to build such kernels to encourage further experiments in other application domains of machine learning.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Similarity measurement processes are a core part of most machine learning algorithms. Traditional approaches focus on either taxonomic ("A and B share properties x, y and z") or thematic ("A is similar to B by value N") thinking. Psychological research, e.g.[WB99], suggests that a combination of both is needed to adequately model human-like similarity perception.

Any model combining those aspects is called a Similarity Dual Process Model. Even though other, unrelated dual processes exist in research, we refer to our Similarity Dual Process Model as DPM for the rest of this work.

The primary aim of this thesis is to provide an implementation of the DPM idea. It should perform binary classification and be adoptable to carry out other machine learning tasks like, for example, cluster analysis, correlation and ranking. The most important question for the implementation is: How can we smoothly integrate a DPM into machine learning algorithms?

The secondary aim of this work is to test DPMs in real-world experiments. The selected scenario should have intermediate applications, while still being simple enough to generate results within reasonable time. The experimental setup should be justified - i.e. it should be clear why which decision was made. In addition, the experiments should be easily reproducible for others.

The question for the experimental results is: Which DPMs performed best? While answering this question, we have to rule out the possibility that our best DPMs are just a product of coincidence and a search space with too many degrees of freedom.

Figure 1.1: Taxonomic and Thematic Thinking (cf. [Eid12, p. 540])

Figure 1.1 tries to develop an intuitive understanding with an example. The triangle on the left side is the reference. We compare it to the two objects next to it. If the focus lies on taxonomic thinking, the triangle in the center is more similar to the reference, because it also has three corners. If the focus lies on thematic thinking, the square on the right is less different from the reference, because it has a smaller height difference.

At first sight, the combination of both concepts seems an unnecessary complication. After all, both have been used on their own in machine learning - mainly in models created by computer scientists. Humans, however, do not always use one or the other approach when making similarity judgments.

One experiment asked participants to rate the similarity of word pairs on a numeric scale, e.g. (milk, coffee), (milk, lemonade), (milk, cow) and (milk, horse). "*As one would expect, similarity ratings for pairs which were highly alignable were reliably higher than for pairs which were poorly alignable. However, contrary to present accounts of similarity, ratings for pairs with preexisting thematic relations were higher than for pairs without preexisting thematic relation.*" [WB99, p. 216–217]

In other words, humans tend to think that a cow is more similar to milk than a horse is to milk, because the associate a cow with giving milk. If machine learning is used to solve a purely "objective" task, for example predicting the number of newspapers to ship to a certain town, then this finding might be irrelevant. If the task is to model human behavior, for example recognizing pedestrians in images, then this finding (and others related to taxonomic and thematic thinking) is highly relevant.

We now have most ingredients for formulating a DPM. One remaining complication is that not all humans use taxonomic and thematic to the same extent. During the rest of this thesis, this issue is dealt with by using an importance of taxonomic thinking factor. The calibration of this factor can be done on a per-person basis with psychological tests.

$$s_{dpm} = \alpha \; s_{taxonomic} + (1 - \alpha) \; g(d_{thematic}) \tag{1.1}$$

Equation 1.1 shows a simple DPM [Eid12, p. 540] with the following parts:

- Similarity s_{dpm}
- Importance of taxonomic thinking $\alpha, 0 \leq \alpha \leq 1$
- Specific taxonomic measure $s_{taxonomic}$
- Specific generalization function g (for converting distance to similarity)
- Specific thematic measure $d_{thematic}$

The specific taxonomic measure, thematic measure and generalization function to be plugged into the equation are the reader's decision. Section 4.6 lists some good options which were found during experimentation. Note that, for the rest of this work, we deal with linear combinations of taxonomic and thematic thinking only. Apart from keeping things simple, there is no other rationale behind this choice. It is entirely possible that, for example, a cubic DPM poses a better choice.

We close this introduction with an overview of the chapters to come. Chapter 2 introduces the reader to the concepts we need later on. This chapter also plays the role of a brief literature survey. DPMs are a novel approach to measuring similarity. Therefore, we discuss other models for measuring similarity that have been proposed. Next, we further explain taxonomic and thematic thinking and two different kinds of measures.

Afterwards, we turn to more technical aspects arising from the real-world task we selected: the detection of pedestrians in images. It was selected, because of its interesting applications (e.g. automatic braking for pedestrians in modern cars, computer games with a new level of interactivity and many more) and because various well-known algorithms already exist for it.

For pedestrian detection, we make use of the image detection algorithm Histogram of Oriented Gradients and two other algorithms from the MPEG-7 standard: Scalable Color Descriptor and Edge Histogram Descriptor. A discussion of Support Vector Machines, which are binary classifiers, concludes the chapter.

Chapter 3 describes how we reached our primary goal. After an overview, we discuss the implementation of pedestrian detection. Next, we turn to the implementation of measures, generalization functions and DPM kernels. The complete code is provided as a free download. The last part of this chapter is a discussion of the experimental setup.

We go through the outcomes of the experiments in Chapter 4. We start with a comparison to existing models and discuss the viability of DPMs. Next, we take a look at the effect of using different generalization functions and different measures. At this point, we are able to list the DPMs that performed best, thereby reaching our secondary goal. Evidence for the statistical significance of the results concludes the chapter.

Eventually, Chapter 5 gives a conclusion and mentions promising areas of further research.

CHAPTER 2

Background

2.1 Similarity

Judging similarity, as a major part of cognition, is very easy for humans. Why, for example, a fir tree is similar to a pine tree, is intuitively clear for us. However, formalizing our similarity judgment process proves to be a very hard task. In the next section, we are therefore going to summarize some influential models for measuring similarity that have been proposed so far.

There is no grand unified theory of the human perception of similarity. The complexity of the proposed similarity models varies greatly, while no single model is general enough to be usable for all tasks.

Similarity is a relationship between objects. This definition seems extremely vacant, but it is a good way to accommodate the numerous uses of similarity. The exact properties of the relationship are not known in general. As we will later see, even properties as likely as symmetry do not always hold, e.g. an apple need not be as similar to an orange as an orange is to an apple. The type of relationship is also not known in general.

Similarity is often represented by a binary relationship. Experiments with groups of identical and differing objects show that this not accurate in all cases (e.g. the experiments carried out on pigeons by Wasserman [WHK95]).

2.1.1 Geometric Models

Geometric models were one of the earliest models for similarity. Despite some shortcomings, they are still widely used today. Reasons for this are their clarity and intuitiveness. The main idea is to place objects in N dimensions and measure their distance with a metric. The arguably most famous metric is the Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}; \quad x, y \in \mathbb{R}^N \tag{2.1}$$

Metrics used in geometric models usually have the properties of a metric function $d : X \times X \rightarrow \mathbb{R}$. We will later take a look at the evidence against these properties $\forall x, y, z \in X$:

- Non-Negativity: $d(x, y) \geq 0$
- Identity: $d(x, x) = 0$
- Symmetry: $d(x, y) = d(y, x)$
- Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

Multi Dimensional Scaling (MDS) is a more advanced geometric approach. It takes any measure of pairwise proximity as input and constructs a geometric model of the data. The distance between every two objects in the geometric model is related to their distance in the input (cf. [GS05, p. 5]).

The main purpose of MDS is to represent proximity data in a readable way. It is often easier to look at a plot rather than all the values in a proximity matrix. Another purpose is data compression.

If we have N objects, we need $N - 1$ dimensions for our MDS to perfectly reconstruct the proximity in the input. However, we can reduce the number of dimensions if we accept some loss of proximity information. Let us look at an example (cf. [GS05, p. 6]). Assume we have the following events:

- $Similarity(Russia, Cuba) = 7$
- $Similarity(Russia, Jamaica) = 1$
- $Similarity(Cuba, Jamaica) = 8$

If we build an one dimensional model, we cannot place Russia near Cuba and far away from Jamaica, because Cuba is near Jamaica. However, if we build a two dimensional model, it is possible. MDS provides information about the cognitive process. Often, the axes can be labeled with interpretations. In our example, the labels are "political affiliation" and "climate".

The goal of MDS is to minimize the badness of fit between the distance of the inputs $\delta_{i,j}$ and the distance of the outputs $||x_i - x_j||$, as shown in Equation 2.2. We could, for example, use a pairwise dissimilarity matrix as input and measure distance between the outputs with Euclidean distance.

$$\underset{x_1,\ldots,x_n}{\arg\min} \quad \sum_{i<j} (||x_i - x_j|| - \delta_{i,j})^2 \tag{2.2}$$

Note that x_i in the solution is not unique. The reason for this is that we can translate, rotate and reflect the points without changing the distance.

2.1.2 Featural Models

Featural Models are motivated by evidence against properties of the metrics used for geometric models. For example, in [PG79], if two identical letters are shown side by side, the reaction time for finding out whether the letters are similar varies. This a violation of the identity property of geometric models.

Using psychological experiments, Tversky ([Tve77]) found a good example for a violation of the symmetry property: North Korea is judged more similar to Red China than Red China is to North Korea.

According to the triangle inequality assumption, in Figure 2.1, the distance between A and C cannot be larger than the distance between A and E plus the distance between C and E.

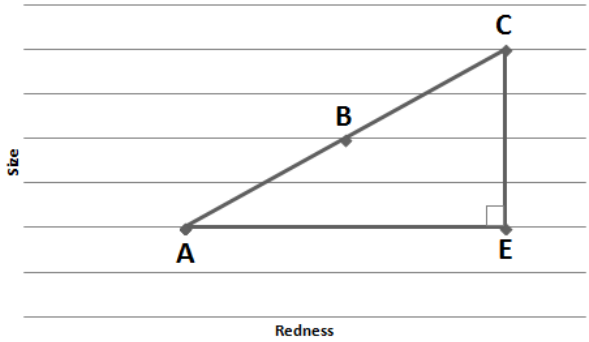


Figure 2.1: Triangle Inequality

According to the segmental additivity assumption, if A, B and C lie on a straight line, then the distance between A and C must equal the distance between A and B plus the distance between B and C.

If segmental additivity is combined with the triangle assumption, and E forms a right triangle when combined with A and C, then one of two cases must hold:

1. The distance between A and E must be larger than or equal to the distance between A and B. Additionally, the distance between E and C must be larger than or equal to the distance between B and C.

2. The distance between A and E must be larger than or equal to the distance between B and C. Additionally, the distance between E and C must be larger than or equal to the distance between A and B.

Again, Tversky carried out psychological experiments and found a violation of the triangle assumption if it was combined with the segmental additivity assumption ([TG82]). Test subjects had to assess the similarity of color/length-pairs to each other. The pairs were chosen according to Figure 2.1, e.g. E had the same size as A and the same redness

as C. People's dissimilarity ratings were not consistent with what we just derived, e.g. the distance between A and E was rated smaller than the distance between A and B.

$$S(a,b) = xf(a \cap b) - yf(a \setminus b) - zf(b \setminus a) \tag{2.3}$$

The contrast model addresses the problems we just described. It was proposed in [Tve77]. The contrast model is defined by Equation 2.3. S and f measure similarity, while x, y and z perform a weighting of the different components. $a \cap b$ denotes the features that are in both a and b. $a \setminus b$ denotes the features that are in a, but not in b. $b \setminus a$ works the other way round.

2.1.3 Transformational Models

The idea of transformational models is that any two representations of objects can be transformed into the other. The easier this transformation is, the more similar the objects are. How the difficulty of the transformation is measured and which transformations are available, depends on the specific model.

We will take the Imai model [Ima77] as an example, which uses strings of Xs and Os as stimuli. Four transformations are possible:

- Mirror image (e.g. $XXXXXOO \rightarrow OOXXXXX$)
- Phase shift (e.g. $XXXXXOO \rightarrow XXXXOOX$)
- Reversal (e.g. $XXXXXOO \rightarrow OOOOOXX$)
- Wave length (e.g. $XXOOXXOO \rightarrow XOXOXOXO$)

The more transformations we need to make two strings identical, the less similar they are. Additionally, strings that can be made identical in multiple ways, are more similar.

Transformational models sometimes lack versatility for practical applications. Let us, for example, compare two digital images with a transformational model. Even with very sophisticated transformation operations, there would always be many possible sequences of transformations, making the time complexity large. In addition, we would often be forced to change the intensity of individual pixels, because no other identity-producing transformation exists.

2.2 Taxonomic and Thematic Thinking

This work is about a rather new model for measuring similarity: the DPM. Its theoretic foundations are taxonomic and thematic thinking, which are introduced in this section.

Taxonomic thinking tries to identify common features between objects. The more common features can be identified, the larger the similarity. Let us, for example, compare a table to a chair. A possible common feature is "number of legs". Because both have four legs, they are similar if we use taxonomic thinking.

Thematic thinking tries to find a theme that connects the objects. This theme is then used for comparison. In our example, a possible theme is "furniture". The likelihood to find a table and a chair at a furniture store is nearly equally high. Therefore, tables and chairs are similar.

The usual definition of a theme is any "temporal, spatial, causal or functional relation between things" [EGJ11, p. 3]. In our introductory Figure 1.1, the theme is size. In other words, the two objects triangle and square are thematically related via their size. Different viewers of the test figure might find different thematic relations in it, e.g. color, occurrence in a company logo and also subjective ones like beauty.

Table 2.1 summarizes the properties of taxonomic and thematic thinking.

Because we only have our senses to experience reality (at least according to a Positivism world-view), objects that we compare are represented as stimuli. Stimuli are said to be separable if they are either represented as 0/1-values (i.e. predicates) or can be counted. Other kinds of stimuli are referred to as integral stimuli. Humans can identify separable stimuli quickly and easily.

If we think back to our introductory example in Figure 1.1, the number of edges was a separable stimulus. while the object size was integral. Separable stimuli are associated with taxonomic thinking, while integral stimuli belong to the world of thematic thinking. We can argue that if we compare images, we have both separable and integral stimuli to compare at the same time. This is further evidence in support of DPMs.

Taxonomic thinking and thematic thinking are not synonyms for similarity and distance. It is possible to measure distance taxonomically and vice versa. We could have said the chair and the table in our example have a "number of legs" difference of zero. Analogously, we could have said that both chair and table have the common feature "sold at furniture store".

However, it makes sense to associate taxonomic thinking with similarity, because similarity measurement works best on separable stimuli. In an analogous manner, distance measurement works best on integral stimuli cf. [Eid12, p. 537].

To measure distance between two concepts, it is often constructive to find a commonality first. We call such concepts alignable. For example, while comparing a car and a motorcycle, we can find the commonality "wheels" and use it to measure distance. Concepts with no meaningful commonalities are called nonalignable.

Highly alignable concepts tend to lead to taxonomic thinking, because there are enough common features to be able to make a comparison. Poorly alignable concepts tend to lead to thematic thinking, because there are no common features to even work with taxonomic thinking. Humans even go as far as inventing unlikely thematic relationships in this case. Distance can be converted into similarity with a generalization function which allows us to mix similarity and distance in DPMs. The exact form of the generalization function is still being disputed.

Depending on whether we measure or count, different measures (for want of a better name) are used for taxonomic and thematic thinking. The dot product and the number of co-occurrences are typical choices for taxonomic thinking. The city block distance and the Hamming distance (features that are in one object, but not the other) are typical

Property	Taxonomic	Thematic
Stimuli	Separable	Integral
Concern	Similarity	Distance
Measurement	Dot Product	City Block Distance
Counting	Co-Occurrences	Hamming Distance

Table 2.1: Properties of Taxonomic and Thematic Thinking (cf. [Eid12, p. 537])

choices for thematic thinking.

2.3 Generalization Functions

We already heard that in nature, similar objects often share the same properties and how this is useful for cognition. This can be formalized with generalization functions. The higher the difference between objects, the lower the probability of them belonging to the same class.

Identical objects should belong to the same class with a probability of $p = 1$. From this point, probability should fall with similarity. How exactly it should fall, is still being disputed. Various generalization functions have been proposed that had varying success with different types of stimuli.

Generalization functions allow us to convert distance into similarity, because we can simply use the probability of belonging to the same class as a measure of similarity. This is an important part of DPMs, because we need a way of combining similarity and distance.

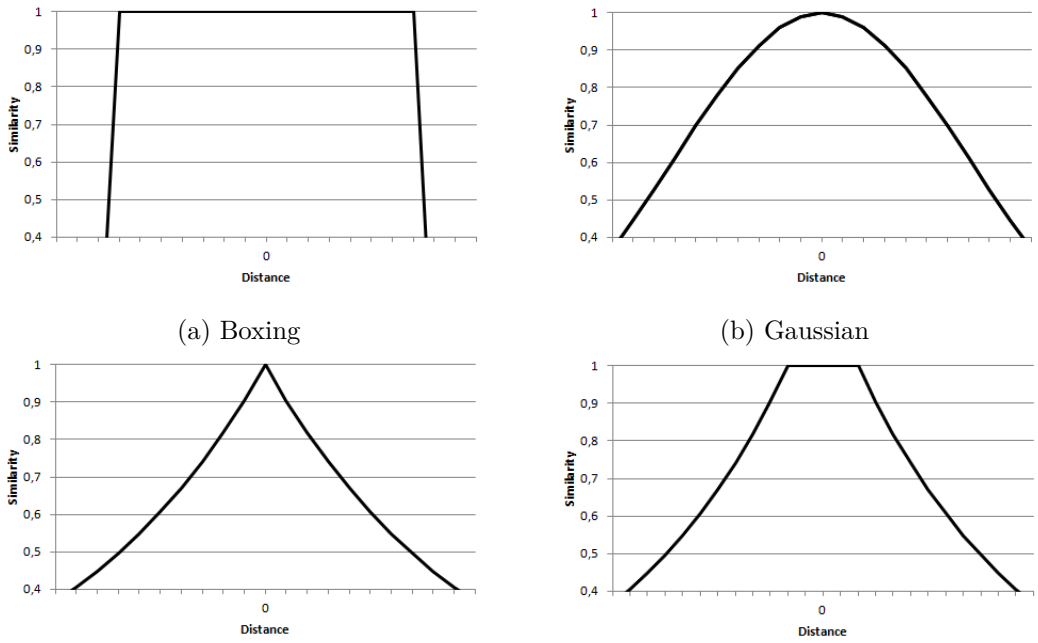
Note that we deal with negative distances with a symmetry assumption to simplify distance measurements. The distance between object x and y yields the same probability as the distance between object y and x.

Figure 2.2a shows the Boxing generalization function. The form that we show allows some distance between objects that are rated as equal.

Figure 2.2b shows the Gaussian generalization function. The basic idea of this function is that the probability of belonging to the same class should decay exponentially with distance.

Figure 2.2c shows the Shepard generalization function. Compared to the Gaussian generalization function, it falls of more steeply. That means that a rise in distance leads to a faster decline in probability. The Shepard generalization function represents Shepard's Universal Law of Generalization ([She87]). It states that the probability that a stimulus is associated with a certain response is proportional to $e^{-distance}$ in an appropriate psychological space.

Figure 2.2d shows the Tenenbaum generalization function. It can be seen as a more general version of the Shepard generalization function, because it is based on Bayesian inference and extends generalization to multiple consequential stimuli. We can see that its peak is broader than the peak of Shepard's generalization function. This is an intuitive



(a) Boxing

(b) Gaussian

(c) Shepard

(d) Tenenbaum

Figure 2.2: Generalization Functions

result of dealing with multiple stimuli. If we get more positive examples for learning, we get surer about the classification of new examples.

Details about the Tenenbaum generalization function can be found in [TG01]. Note that, because the concrete size of the peak depends on the concrete series of stimuli, the Tenenbaum generalization function is logically very similar to our Boxing generalization function.

2.4 Measures

We already mentioned that we need measures for our DPM in Equation 1.1. In the following section, we are going to introduce two classes of measures. The distinction is made on a rather technical basis: the function signature. Bear in mind that this is not related to the distinction into taxonomic and thematic thinking, which is discussed in Section 2.2.

2.4.1 Quantitative Similarity Measures

The parameters of quantitative measures are at least interval-scaled. In simple terms, this means that they can be compared and subtracted, while divisions need not be possible. The operation results are not required to be interval-scaled themselves. In even simpler terms, quantitative similarity measures take real numbers as inputs.

We already introduced Euclidean distance in Equation 2.1. It is a quantitative measure for distance and therefore associated with thematic thinking. Another classical example for a quantitative measure is the dot product defined in Equation 2.4. Unlike Euclidean distance, it measures similarity and is therefore associated with taxonomic thinking.

$$s(x,y) = \sum_{i=1}^{N} x_i y_i; \quad x,y \in \mathbb{R}^N \tag{2.4}$$

2.4.2 Predicate-based Similarity Measures

Predicate-based measures work with nominal parameters. Because one predicate is as good as the other, it does not make sense to work with them individually. Therefore, two description vectors f_1 and f_2 are combined into variables:

- a counts the number of predicates that are present in both description vectors
- b counts the number of predicates that are present in f_1, but not in f_2
- c counts the number of predicates that are present in f_2, but not in f_1
- d counts the number of predicates that are present in neither description vector

Variables a, b, c and d are the parameters of a predicate-based similarity measure. As before, there is no need for the result to be interval-scaled. Also note that with K the number of feature vector elements, $d = K - a - b - c$. Predicate-based measures are allowed to ignore some of the variables. d is not used in many measures.

$$d(a,b,c,d) = b + c \tag{2.5}$$

Let us conclude this section with two examples. The Hamming distance in Equation 24 focuses on the differences between two feature vectors and on thematic thinking. With $f_1 = \binom{1}{1}{0}$ and $f_2 = \binom{0}{1}{0}$, we get $a = 1$, $b = 1$, $c = 0$ and $d = K-a-b-c = 3-1-1-0 = 1$ and eventually, $d = 1$.

The number of co-occurrences in Equation 2.6 focuses on similarity and taxonomic thinking. With our feature vectors from before, $s = a = 1$.

$$s(a,b,c,d) = a \tag{2.6}$$

2.5 Feature Extraction

In his seminal article [Att54], Attneave argues that visual information reaching the eye is highly redundant and calls for a more economical representation of this information. He also states that attaining this compression is one of the major goals of visual perception.

Figure 2.3: Contours of a Sleeping Cat (cf. [Att54, p. 185])

Figure 2.3 shows an example for compression. The drawing was made by abstracting 38 points of maximum curvature and connecting them with appropriate straight lines.

We can see that, despite a lot of information being lost during the compression, the sleeping cat is still clearly recognizable. The points of maximum curvature can be seen as a summary of the original media object that is still detailed enough for the task at hand.

We call such summaries features and the process of generating them is called feature extraction. While data compression is beneficial, it is not the most useful impact of feature extraction: Feature vectors make objects comparable. For example, if we compare two digital images bit by bit, we will only get a good similarity assertion if they are exactly equal.

Even the slightest difference in camera angle, lighting, etc. could lead to very different binary representations and thus to unusable similarity judgments. By using feature vectors as an intermediate format, our similarity judgments become more robust.

We used image feature extraction as an example, but the idea is not restricted to images. Many algorithms exist for video and sound feature extraction. The concrete algorithm depends heavily on the issue to be solved. It is a reasonable assumption that, in the future, a feature extraction algorithm is found that works equally well for all types of media objects and all domains. Humans are able to make similarity judgments regardless of stimulus type, so this grand goal should be realizable.

Usually, features span more than one dimension. The cat in our example could be identified by the Cartesian coordinates of the points in the precise order that lines should be drawn between them. Data like these are best stored in a vector with numeric elements that we call a feature vector.

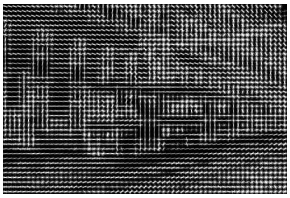
Usually, feature vectors are compared element-wise. This causes problems if two feature vectors have different dimensions or the objects they summarize have different dimensions. Most feature extraction algorithms always produce feature vectors of equal length - sometimes simply by filling missing dimensions with zeros.

The problem of different input dimensions can be solved by splitting the input into equal sizes or by using an algorithm that is independent of the input dimensions.

In the following section, we are going to learn a concrete algorithm to extract feature vectors from images.

(a) Input Image

(b) Image Gradient



(c) HOG Visualization

Figure 2.4: HOG Algorithm Stages

2.6 Histogram of Oriented Gradients

HOG divides images into parts and creates a histogram for each part. The histograms describe the gradient orientations of each part, which can be calculated from the horizontal and vertical gradients. HOG does not know about the the concrete edge positions - it is only concerned with the distribution of gradient orientations in the histograms (cf. [DT05, p. 2–3]).

For a task like human detection, the high-level approach of HOG is well suited, because depending on body position, camera position and lighting, humans look very differently on images. The exact location of all body parts would be too much effort for a binary classification, while HOG can be calculated efficiently and robustly. The simplified steps of the HOG algorithm from input image to person/non-person classification are:

1. Computation of gradients
2. Voting into spatial and orientation cells
3. Normalization over overlapping blocks
4. Support vector machine classification

The original HOG paper [DT05] explores many possibilities of implementing the algorithm. To keep things simple, we will describe one variant with good detection performance here and visualize the steps with the HOG Glasses project[1]. Figure 2.4a shows our input image.

First, gradients are computed with the centered 1-D masks $filter_h = (\,-1\ 0\ 1\,)$ and $filter_v = (\,-1\ 0\ 1\,)^T$. Gradient calculation is done for each color channel, but only the gradient vector with the largest norm is used. Figure 2.4b shows an example for a gradient.

Second, each pixel casts a vote with its gradient magnitude for a histogram bin. The bins are evenly spaced over 0° and 180° into 9 parts. The gradient orientation determines the bin to vote for. A cell is a small subpart of an image. Each cell has its own histogram. Anti-aliasing is done between neighboring bins.

[1] `http://saturday.csail.mit.edu:8080` last accessed 2015-02-09

Third, the cells are grouped into larger units: the so called blocks. Blocks are rectangular and overlapping, so that each cell is part of more than one block. The concatenation off all histograms within a block forms a vector v, which is normalized with the L2-norm $v_{normal} = \frac{v}{\sqrt{||v||_2^2+\epsilon^2}}$, ϵ being a small constant. Figure 2.4c visualizes this step.

Finally, a SVM with a linear kernel is trained with HOG descriptors. The resulting SVM model can then be used to classify new images. More information about SVM can be found in Section 2.8.

2.7 MPEG-7 Visual Standard

In this section, we will describe two additional ways of extracting feature vectors taken from the MPEG-7 Visual standard. In principle, HOG alone is enough to produce satisfactory results. However, HOG creates quantitative feature vectors with many decimal places because of the normalization step. The integral numbers created by the two following MPEG-7 descriptors are much easier to turn into predicates and allow us to test predicate-based measures, too.

The aim of the MPEG-7 Visual standard is to provide standardized descriptions of images and video. The descriptions alone are not very useful, but they help us to identify, categorize and filter media objects. MPEG-7 Visual should make it easier to implement multimedia applications (cf. [Sik01, p. 696]).

MPEG-7 descriptors can be classified into general and domain-specific descriptors. This work deals with the following general descriptors, that will be used during the implementation in Chapter 3:

1. Scalable Color Descriptor (SCD): Creates a histogram of the color distribution of an image

2. Edge Histogram Descriptor (EHD): Captures the spatial distribution of the edges of an image

2.7.1 Scalable Color Descriptor

SCD divides the HSV (hue-saturation-value) color space into 256 bins of uniform size. This includes bins for hue, saturation and value. Each pixel is quantized into its nearest hue, saturation and value bins. This is called a color histogram. Algorithm 2.1 shows a simplified algorithm for its generation.

The histogram values are truncated to 11-bit integers and nonlinearly mapped into a 4-bit representation. This step makes small values with a low probability of occurring in an image more significant. At this point, the descriptor can be used, although there is one drawback: its space requirements, which is also related to the time it takes to compare two descriptors. This is important, because hardly any practical tasks need just one descriptor without performing some comparison process.

To make the SCD smaller, a Haar transformation is employed (cf. [OCK01, p. 9]). Neighboring bin values are combined into one bin value, which from then on represents a Haar coefficient. Each pass halves the number of bins and therefore the histogram size. SCD derives its name from the fact that different compression levels can be compared to each other.

Usually, SCD-comparisons arise from image retrieval tasks. The specification suggests to use the L1-norm in the Haar domain or directly in the histogram. In Chapter 3, we will also explore the use of other measures in the latter domain.

Algorithm 2.1: Example for Generating a Color Histogram

Input: h, s, v HSV image channels
Output: *hist* color histogram

```
1 while x=h->next() && y=s->next() && z=v->next() do
2 |   ++hist[getNearestBin(x, HUE_BIN_SIZE)];
3 |   ++hist[getNearestBin(y, SATURATION_BIN_SIZE)+OFFSET1];
4 |   ++hist[getNearestBin(z, VALUE_BIN_SIZE)+OFFSET2];
5 end
```

2.7.2 Edge Histogram Descriptor

EHD searches for edges in an image and puts information about them in a histogram. The image is divided into 16 identically-sized blocks. Each block occupies 5 bins in the histogram: 0°(horizontal), 45°, 90°(vertical), 135°and isotropic (not oriented). The blocks are further subdivided until a level of $2x2$ pixels is reached.

On this micro-level, simple edge detection operators are applied to the intensity values, e.g. the operator for the 45°-bin $op_{edge} = \begin{pmatrix} 0 & \sqrt{2} \\ -\sqrt{2} & 0 \end{pmatrix}$. If the result of the maximal operator exceeds a certain threshold, the average intensity value of the micro-level is added to the appropriate bin. Finally, the bin values are normalized and compressed to 3 bits to keep descriptor size low.

As before, the specification suggests using the L1-norm for EHD-comparisons - this time directly with the 3 bit bins. Again, we are going to explore the use of other measures in Chapter 3.

2.8 Support Vector Machines

Now that we know how to transform an image into a feature vector, we return to our binary classification task. We could directly calculate distances between a known positive feature vector (e.g. contains a pedestrian) and an unknown feature vector. But with this approach, we have to set an arbitrary boundary and the classification performance would not be very high. Therefore, we introduce a classic machine learning algorithm for binary classification: the Support Vector Machine.

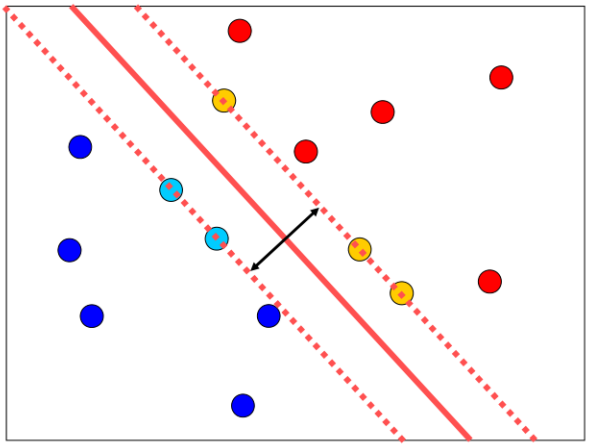


Figure 2.5: Principle of a SVM (cf. [Bis06, p. 327])

2.8.1 Overview

SVMs try to separate data by drawing a line between the two classes. If the data are separable in this way, then there are usually many ways to draw this line. The question is: Which line to take? Some algorithms, like the Perceptron algorithm, just take the first one they find. SVMs answer this question by maximizing inter-class distance.

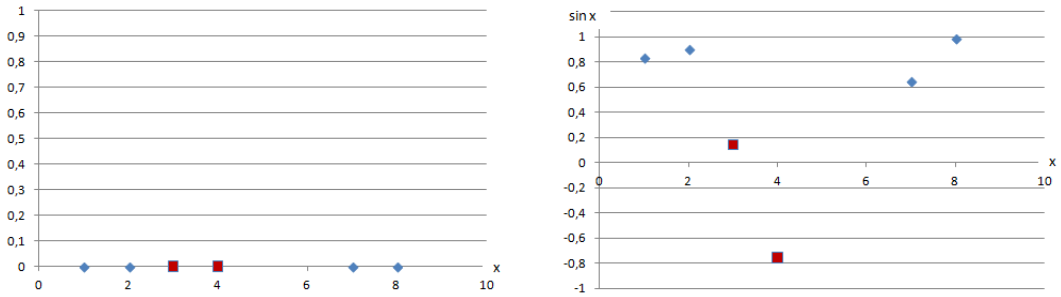
The rationale of SVMs is to place this line (i.e. decision boundary) as far away from both classes as possible. This reduces the misclassification risk of new data points. The principle of the SVM algorithm can be seen in Figure 2.5. The light and dark blue points represent class A, while the orange and red points represent class B. The solid red line is the decision boundary. New data will be classified as A if it is below the boundary and as B otherwise.

The optimization goal of the algorithm is to select the data points that maximize inter-class distance and therefore span the classification boundary. Our image shows them in light blue and orange. Vapnik, inventor of the first SVM, gives a good introduction in [Vap00].

SVMs are defined by the linear model in Equation 2.7. Parameters $\vec{w}^T$ and b make up a SVM model. N training data points are stored in $\vec{x_1}, ...\vec{x_N}$ with ground truths in $t_1, ..., t_N$ where $t_n \in \{-1, 1\}$ are the class labels.

$$y(\vec{x}) = \vec{w}^T\vec{x} + b \tag{2.7}$$

A detailed, step-by-step solution is out of scope for this work. It is provided, for example, in [Bis06, p. 326–336]. We find the parameters by formulating the problem in Figure 2.5 as a constrained optimization problem. We use the Lagrange approach and work with the dual form of the problem. Finally, the Karush-Kuhn-Tucker conditions help us to find a solution with quadratic programming.



(a) Not Possible (b) Possible

Figure 2.6: Linear Class Separation

2.8.2 Kernels

$$L_{dual} = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m t_n t_m x_n x_m \tag{2.8}$$

The Lagrange polynomial needed for finding the SVM parameters is given in Equation 2.8. It has the useful property that the data points x_j appear only as pairs linked by multiplication - i.e. as a dot product. We already defined the dot product in Equation 2.4. It also has a geometric definition with θ the angle between two vectors, given by Equation 2.9.

$$s(\vec{x}, \vec{y}) = ||\vec{x}|| \; ||\vec{y}|| \; cos(\theta) \tag{2.9}$$

If we think about the dot product like this, we see that it is a very natural similarity measure. It measures directional similarity of two vectors. Let us look at an example of two orthogonal vectors. Because $cos(90°) = 0$, their similarity is 0. If we have parallel vectors $\vec{x}$ and $\vec{y}$, then, because of $cos(0°) = 1$, their similarity is $||\vec{x}|| \; ||\vec{y}||$.

In Equation 2.8, we factor out $x_n x_m$ and replace it by some function $k(\vec{x}, \vec{y})$ that we call a kernel function. Any function can be used as a kernel function, as long as it is symmetric and positive semidefinite. There is evidence that even the definiteness requirement for kernel functions can be violated, e.g. [OMCS04].

This replacement is called kernel trick. It is an interesting idea for machine learning and works with any algorithm for vector data that can be expressed in terms of dot products between vectors. The data is transformed to a higher-dimensional space to make separation easier.

Often, data is not linearly separable in the original space. We can see an example for this in Figure 2.6a. One possibility is to allow some separation errors and include this into the optimization problem with a penalty function. Another approach is to use a kernel function that maps data to a higher dimensional space where the separation is possible. An example for this is shown in Figure 2.6b.

Using kernel functions is a simple and efficient way to add non-linearity to linear algorithms and as such not restricted to SVMs. The simplest SVMs are linear algorithms,

in the sense that they try to find a linear separation in the input space. If input space is transformed to a higher-dimensional feature space with the kernel trick, then we can add nonlinearity to the SVM algorithm without changing it and without additional computational cost (cf. [VTS04, p. 12]).

Another useful aspect of the kernel trick is that we are not required to explicitly know the function that maps the original data to higher-dimensional feature space. This is not a problem, because there is no need to apply this mapping function to the original data. It is sufficient for us to be able to compute distance in the higher-dimensional space. This distance alone can often be computed very efficiently.

Kernel functions can be visualized by plotting them against the similarity of two vectors. Notable kernel functions are:

- Linear kernels $k(\vec{x}, \vec{y}) = \vec{x}'\vec{y}$: They are equivalent to not performing a kernel substitution and creating no additional dimensions. Similarity is measured with the dot product. An example for a linear kernel is shown in Figure 2.7a.

- Gaussian kernels $k(\vec{x}, \vec{y}) = e^{-a(x-y)^2}$: They do not create additional dimensions. The gestalt of these kernels is very similar to the Tenenbaum and to the Shepard generalization function that we can use in the thematic part of our DPM. They measure distance at first, but convert it into similarity with exponential decay. An example for a Gaussian kernel is shown in Figure 2.7b.

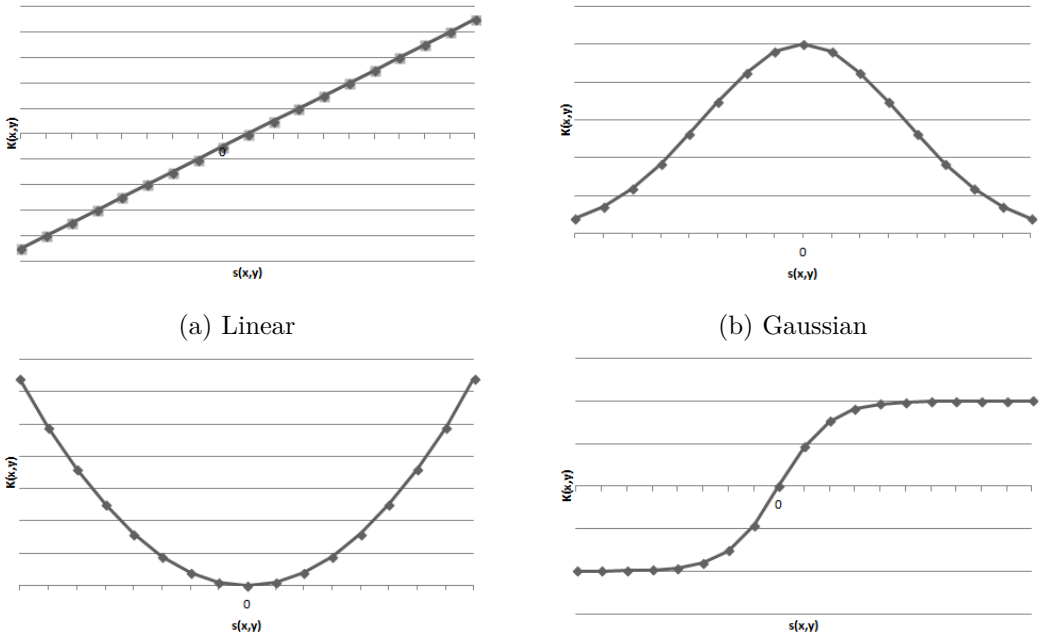
- Polynomial kernels $k(\vec{x}, \vec{y}) = (1 + \vec{x}'\vec{y})^a$: They create a dimensions to better separate data. Again, similarity is measured with the dot product. An example for a polynomial kernel is shown in Figure 2.7c.

- Perceptron kernels $k(\vec{x}, \vec{y}) = tanh(w_0\vec{x}'\vec{y} + w_1)$: They imitate a neuron that fires until it has reached its highest potential. They measure similarity. An example for a Perceptron kernel is shown in Figure 2.7d.

Remember that the motivator for this thesis is the insight that the traditional, either exclusively taxonomic or exclusively thematic, approaches for similarity models are not very close to the way humans measure similarity. Kernels can be thought of as similarity measures. They become large if two vectors are similar and small if they are dissimilar (cf. [VTS04, p. 6–7]).

Existing kernels usually neglect thematic or taxonomic thinking. For example, consider the linear kernel of a SVM. It is equivalent to the dot product, which is a similarity measure and therefore concerned with taxonomic thinking. Our idea is to replace the similarity measurement done by a kernel with a dual process model that combines taxonomic and thematic thinking.

Because we need similarity (related to taxonomic thinking) in the end, we introduce difference (related to thematic thinking) by converting it into similarity with a generalization function. Therefore, we can just use our DPMs as kernels, i.e. we can set $k = m_{dpm}$. The DPM kernels in this work usually only perform similarity measurement



(a) Linear

(b) Gaussian

(c) Polynomial

(d) Perceptron

Figure 2.7: Kernel Functions

and no mapping. One exception appears in Equation 3.3, but we excluded it from the experiments to keep the number of tested DPMs manageable.

In conclusion, this chapter introduced some of the models for measuring similarity that have been proposed. We explained taxonomic and thematic thinking, because these aspects are often not accounted for in similarity models. We took a look at generalization functions, because they enable us to mix similarity and difference. The distinction between quantitative and predicate-based measures was explained because both kinds can be used in a DPM. We also covered the technical aspects that we need later in this work to test DPMs in a real-world situation: Feature extraction in general, concrete feature extraction algorithms, support vector machines and kernels. This concludes the discussion of the background.

CHAPTER 3

Implementation

3.1 Overview

The most important outcome of this thesis is a working implementation of the DPM idea. It should be straightforward to use as a basis for other works. To meet this requirement, the implementation is divided into the two separate parts pedestrian detection and SVM.

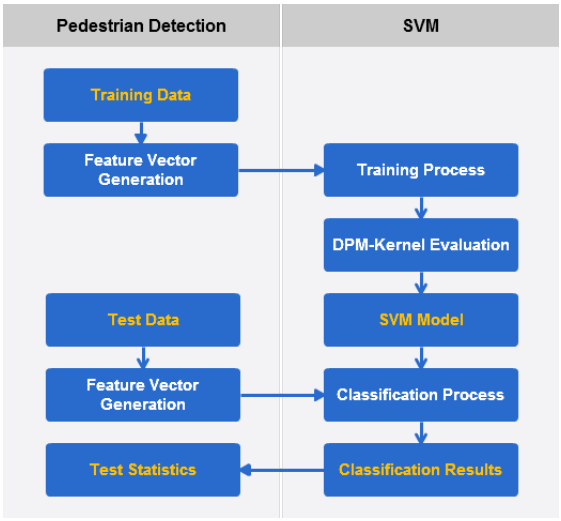


Figure 3.1: Implementation Overview

Pedestrian detection deals with all the domain-specific problems and keeps track of the results. There is not much novelty value in this part of the implementation. Myriads of sample code for this task are freely available[1]. As feature extraction is not the focus of this work, wherever possible, we delegate work to existing libraries.

[1] e.g. `http://iica.de/pd/demos.py` last accessed 2015-02-15

However, understanding and controlling the parts done by libraries, as opposed to e.g. calling a black box function `DetectPedestrians()`, proved surprisingly intricate. The main practical problem was to find a library that does all the involved steps separately and comprehensibly. This problem has been solved eventually.

The second part, SVM, deals with all the domain-independent issues. It has a higher novelty value than the other part, because, to the best of the author's knowledge, no existing classification algorithm implementation supports DPMs out of the box.

The SVM part is independent of pedestrian detection and not even restricted to media descriptions. Although we only tested its performance with pedestrian detection, it is likely to have the same applications as classical SVMs.

Figure 3.1 gives an overview of the implementation. The most important program activities are shown in white, while the most important result files are shown in yellow. The implementation reads training data in the form of images and generates feature vectors from them. These are used to train a SVM.

While training, the SVM uses a DPM kernel instead of a normal one. The result of the training is a SVM model file. At this point, we can discard the training data and start working with separate test data - also in the form of feature vectors. The classification process tells us which images from the test data contain pedestrians and which do not. We end up with a file with the classification results and a file with test statistics.

The complete source code of the implementation is available freely[2].The rest of this chapter takes a more detailed look at the two major parts of the implementation. Next, we explain how to work with the implementation. Treatise about how the experiments were carried out concludes the chapter.

3.2 Pedestrian Detection

Enabling machines to "see" humans is a promising task with a large number of practically very relevant applications like robotics, surveillance and content retrieval. A large amount of research has been conducted in this area and a respectable number of algorithms exists. However, no near perfect algorithm has been proposed so far.

The aim of this work is not to come up with a new, high-performing image detection algorithm. Rather, the effect of using a DPM for measuring distance in existing algorithms is examined. We selected the HOG algorithm (as described in [DT05]), because it is representative for a line of research called gradient-based algorithms.

Empirical research suggests to combine HOG with other feature extraction methods to obtain a stronger algorithm. "*While no single feature has been shown to outperform HOG, additional features can provide complementary information.*" [DWSP12, p. 10]

The Scalable Color Descriptor (SCD) and Edge Histogram Descriptor (EHD) from MPEG-7 Visual provide this additional information to our implementation. Furthermore, turning these descriptor values into predicate based data is straightforward.

SCD and EHD do not return predicates (i.e. zero/one values). To be able to work with predicates, the values returned by SCD and EHD are put into evenly sized bins

[2] `https://code.google.com/p/dual-process-model` last accessed 2015-02-23

by Algorithm 3.1. Note that the algorithm does something different than creating a histogram. The output is an array of values that are either zero or one.

Algorithm 3.1: How to Turn Quantitative Values Into Predicates

Input: *binSize* size of one bin, *minT* smallest possible value, *binNum* number of bins to create, *values* array of values to be binned
Output: *binnedValues* array of binned values

```
1 for i=0; i < values.size(); ++i do
2 |   for b = 0; b < binNum; ++b do
3 |   |   val = values.at(i);
4 |   |   if val >= minT + b * binSize && val < minT + (b+1) * binSize then
5 |   |   |   binnedValues[i*binNum+b]=1;
6 |   |   end
7 |   end
8 end
```

The whole pedestrian detection part has been written in C++. Image loading and HOG feature vector calculation is done using the Open Computer Vision library (Open CV)[3]. A good introduction to the library features can be found in [PBKE12].

OpenCV comes with classes to compute HOG feature vectors and with classes to work with support vector machines. The latter ones have not been used on purpose. The rationale is that machine learning is not a core part of OpenCV. It is there, because some computer vision algorithms need it. For solving a pure machine learning task, OpenCV is not the best choice because of the library size and numerous dependencies which are not even needed for machine learning.

Therefore, and to make it easier for users from non-computer vision domains, another library has been used for the machine learning part. Refer to Section 3.3 for details. Similarly, MPEG-7 feature vector calculation is done using the BilVideo-7 project (see [BcGOU09]).

The pedestrian detection part first reads all training images with pedestrians (positive images) from some directory. Training images without pedestrians (negative images) are taken from another directory. There are virtually no constraints for the number of images, but too few or too many images might yield a poorly trained classifier, because of under-/over-fitting. Experience showed that 20 to 400 training images usually lead to good classification results.

The first program version kept all images and feature vectors in memory. This limited the amount of images that could be processed. The current version frees images and feature vectors from main memory as soon as possible. Furthermore, all data is kept in temporary text files. With this approach, memory usage is nearly independent of the number of images.

[3] `http://opencv.org` last accessed 2015-02-09

There is no need for the user to delete these files manually. The files are stored in directory `gen`. The most important files are the training data (`feature.txt`), the SVM model (`svm.txt`), the test data (`test.txt`), the classification results (`clz.txt`) and the test statistics (`pr*.txt`).

First, the program reads all training images. Not all image detection algorithms can process files with arbitrary size. This ability of an algorithm is called scale invariance. HOG, in the version we use, is not scale-invariant, while EHD and SCD are scale-invariant. Therefore we have to "resize" our images to the correct size.

After this step, feature vectors for training are available. Every feature vector starts with the class label. For us, 1 means the file contains at least one pedestrian, while -1 means there is no pedestrian in the image. For test files, 0 is used as a label if the class is currently unknown. The class label is followed by the binned scalable color descriptor, binned edge histogram descriptor and finally the unchanged HOG descriptor. Every feature vector element except the class label carries a one-based index.

We train a modified SVM with the generated feature vectors. This results in a SVM model file containing the support vectors that create an optimal separation of the training data. If we do not want to carry out the training ourselves, we could get similar support vectors from the pre-trained SVM for pedestrian detection that is part of OpenCV.

The pedestrian detection part reads all test images in the same way as the training images. As soon as this is done, the image classification starts. This part is not computationally expensive anymore. We keep track of the classification result for each image and create a file with test statistics in the end.

We use one of the most straightforward methods for evaluating classifier quality: the correct classification rate. More advanced evaluation methods exist. One of them is analyzing precision and recall.

We calculate precision and recall as shown in Equations 3.1 and 3.2. TP (true positives) is the number of positive images with a positive classification result. FP (false positives) is the number of negative images with a positive classification result. FN (false negatives) is the number of positive images with a negative classification result.

$$precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$recall = \frac{TP}{TP + FN} \tag{3.2}$$

Precision measures how relevant our results are. For example, in an image retrieval task, if we search for "pedestrians" and get ten images of pedestrians and nothing else, then our precision is as high as possible. Yet, we do not know how effective our algorithm is. This is measured by recall. Continuing with our example, if there were thousands of pedestrian images and our algorithm found only ten, then it had a low recall.

Sometimes, machine learning algorithms can be improved by setting a better threshold. We already described briefly how a SVM works. From the optimization goal, we can deduce that the threshold with the most correct classifications is $\Delta b = 0$. It is not sensible to

analyze the sensitivity of the correct classification rate to classification threshold changes, because we already know the optimal threshold.

However, the sensitivity of precision and recall to threshold changes is relevant. If we, for example, wanted to be sure to find all images of pedestrians, we could "lower our standards" and set the limit for positive classification to something lower than zero. If we, to formulate another example, wanted to avoid false positives as much as possible, we could "rise our standards" and set the limit to something much higher than zero.

To analyze the sensitivity of precision and recall to classification threshold changes, we just vary the threshold, redo classification and calculate new precision and recall values. The result of this analysis is called a precision/recall-curve. The test statistics files contain such a curve.

Figure 3.2 shows an example for such a precision/recall-curve. The first column in the test statistics file is the threshold used, the second column is the resulting precision and the third the resulting recall. In our implementation, the threshold always runs from -3 to 3 and is increased in 0.1-steps.

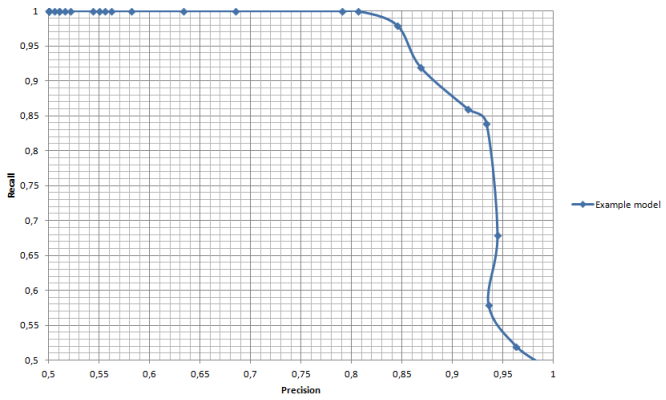


Figure 3.2: Example for a Precision/Recall-Curve

The precision/recall-curves in our implementation allow us to analyze DPM kernel behavior in special tasks where high precision or high recall is required. Precision/recall-curves usually resemble stairs, because there is a tradeoff between precision and recall.

This finishes our tour through the pedestrian detection part. We will now continue with the SVM part. It is important to understand that the pedestrian detection part includes all the "glue code". That means that program execution starts in this part and uses the SVM part for training and classification. When this is done, the pedestrian detection part collects the classification results and puts them in a statistic. The SVM part can still be used standalone as a classification algorithm.

3.3 Classification by a SVM

It makes little sense to implement a whole new library every time an algorithm is modified. Therefore, we built DPM-capabilities into SVM^{light}[4]. It is an implementation of the original SVM idea by Vapnik with numerous improvements.

SVM^{light} is widely used in academia because it is fast, free and easy to modify. The only requirement for using it is quoting [Joa98], which conveniently provides a good overview of the library.

SVM^{light} consists of a learning module and a classification module, both implemented in C. The binary `svm.exe` acts as a front end to both. Custom kernels, like our DPM kernels, are added in function `custom_kernel` in file `kernel.h`.

In Section 2.8, we already heard about kernel functions in general. Equation 1.1 describes the from of our new kernel function. The remaining questions is: Which functions do we plug in? A turnkey list of quantitative and predicate-based measures was found in [Eid12, p. 587–591]. Our DPM kernel supports all measures in this list. Measure indexes are consistent with the numbers from the source, as long as one subtracts 1 from them, because C-arrays are 0-based.

We need generalization functions for our model as well. The implemented generalization functions are Boxing, Gaussian, Shepard and None. They can be used by passing the program one of the indexes $\{0, 1, 2, 4\}$. The missing index has been reserved for the Tenenbaum generalization function that has not been implemented yet.

This freedom to select all parts of the DPM kernel from a list leads to a large number of possible kernels. One of the goals was to make the implementation as dynamic as possible. Therefore, the existing SVM^{light} parameters are "misused" to select and control a DPM kernel.

The disadvantage of this approach is that the parameter names do not have any deeper meaning in the DPM-context. However, the large advantage is that our modified version of SVM^{light} stays fully compatible with the canonical version. This approach is also suggested in the documentation.

DPMs stipulate the use of quantitative and predicate-based measures to represent taxonomic and thematic thinking. We do not mandate which type of measure to use for which type of thinking. We can combine a quantitative measure for taxonomic thinking with a predicate-based measure for thematic thinking or we can use only quantitative or only predicate-based measures.

The concrete combination can be one of the following:

- Quantitative measure for taxonomic thinking. Predicate-based measure for thematic thinking. Selected with index 0.

- Quantitative measure for taxonomic thinking. Quantitative measure for thematic thinking. Selected with index 1.

[4] `http://svmlight.joachims.org` last accessed 2015-02-09

- Predicate-based measure for taxonomic thinking. Predicate-based measure for thematic thinking. Selected with index 2.
- Predicate-based measure for taxonomic thinking. Quantitative measure for thematic thinking. Selected with index 3.

Furthermore, the implementation supports a polynomial DPM kernel that is shown in Equation 3.3. It supports the same combinations as before with the indexes $\{4, 5, 6, 7\}$.

$$s_{dpm_{poly}} = [1 + \alpha \; s_{taxonomic} + (1 - \alpha) \; g(d_{thematic})]^3 \tag{3.3}$$

The pedestrian-detection part always produces feature vectors with both quantitative data and predicates. The custom SVM needs to know which parts of the input are quantitative and which are predicative. This is done by setting the directive `FIRST_N_PREDICATES`. It means that the first N elements of each feature vector in the input should be treated as predicates.

It would not pose a problem to detect the feature vector type automatically. But we are implementing a kernel function - which could well be called thousands of times while searching for a solution. Therefore, we have to make sure it terminates quickly by removing any overhead like this automatic type detection.

Internally, measures and generalization functions are stored in function pointer arrays. `q` is an array of quantitative measures, while `p` is an array of predicate-based measures. Analogously, `g` is an array of generalization functions. The parameters supplied by the users then select the correct DPM kernel parts from the array. Algorithm 3.2 shows a prototype of this concept.

Algorithm 3.2: Prototype for Custom Kernel Selection

Input: *svm_light_parameters* arguments passed to the SVM, α importance of taxonomic thinking, a, b feature vectors
Output: *result* DPM similarity score for feature vectors

```
1  lhs = svm_light_parameters->coef_lin;
2  rhs = svm_light_parameters->poly_degree;
3  combination = svm_light_parameters->coef_const;
4  gi = svm_light_parameters->rbf_gamma;
5  if combination==QUANTITATIVE_QUANTITATIVE then
6  |  result = α * q[lhs](a, b) + (1-α) * g[gi](q[rhs](a, b));
7  end
8  else if combination==PREDICATE_QUANTITATIVE then
9  |  result = α * p[lhs](a, b) + (1-α) * g[gi](q[rhs](a, b));
10 end
11 //...other selections
```

The implementation of quantitative and predicate-based measures is straightforward. Algorithms 3.3 and 3.4 show example measure implementations. Note that, from a purely

Algorithm 3.3: Example Implementation of a Quantitative Measure

Input: a, b feature vectors
Output: *result* Euclidean distance

```
1  register double sum = 0;
2  register WORD *ai, *bj;
3  register int k = 0;
4  ai = a->words;
5  bj = b->words;
6  while ai->wnum && bj->wnum do
7  |   sum += pow(fabs(ai->weight - bj->weight), 2);
8  |   ++k;
9  |   ++ai;
10 |   ++bj;
11 end
12 result = sqrt(sum / k);
```

Algorithm 3.4: Example Implementation of a Predicate-based Measure

Input: a common features in both vectors, b features only in the one vector, c features only in the other vector, d features in neither vector
Output: *result* Hamming distance

```
1  result = b + c;
```

technical point of view, the distinction into thematic and taxonomic measures is not relevant, because they are implemented in the same way.

Quantitative measures are called with the feature vectors in linked lists. Predicate-based measures are a little different. Because they are based on common features (Section 2.4.2), they are already invoked with variables that describe them.

3.4 Usage

The binary file `dpm.exe` performs pedestrian detection with a customized SVM that employs a DPM kernel. The usage is outlined in Algorithm 3.5. It carries out all necessary steps for the training of a support vector machine for pedestrian detection and keeps track of the results. It operates in so called "modes":

1. `features`: Calculates feature vectors.
2. `train`: Uses the features vectors to train a support vector machine.
3. `classify`: Performs classification.

If no mode is given, all steps are performed in succession. This is the usual program invocation, unless one wants to examine the intermediate results or keep track of program

output. Passing parameters to SVM^{light} only makes sense in mode `train`, because before it, no SVM is used and after it, a SVM model file already exists from which the parameters are taken.

Algorithm 3.5: Program Invocation

```
1 dpm.exe [mode] [svm-light-parameters]]
```

The binary passes any parameters other than the first to SVM^{light}. This is used to select the specific DPM kernels. If we do not pass any parameters, we get a linear kernel. To work with DPM kernels, the parameter $-t\ 4$ is needed to tell SVM^{light} to use the custom kernel, which is exactly where we placed all the DPM logic. In this case, some of the standard parameters change their meaning:

- `-g [0...4]` Index of generalization function (refer to Sections 2.3 and 3.3).
- `-s [0...74]` Index of taxonomic measure (refer to Sections 2.2 and 3.3).
- `-d [0...74]` Index of thematic measure (refer to Sections 2.2 and 3.3).
- `-r [0...7]` Index telling us how to combine the measures (refer to Section 3.3).

We implemented 75 predicate-based measures and 19 quantitative measures. Note that because we can theoretically use predicate-based measures for both the thematic and the taxonomic part of our DPM, both indexes run to 74. If a quantitative measure is used, it is important to not exceed the index 18.

We conclude this section with the usage example shown in Algorithm 3.6. It employs the Boxing generalization function to combine the predicate-based measure Mc Connaughey for taxonomic thinking with the predicate-based measure Hamming distance for thematic thinking. To demonstrate compatibility, we also use a standard SVM^{light} parameter to limit the number of optimizations to 10000.

Algorithm 3.6: Example Usage

```
1 dpm.exe features
2 dpm.exe train "-t 4 -g 0 -s 33 -d 2 -r 2 -# 10000"
3 dpm.exe classify
```

3.5 Experiments

DPMs can work with arbitrary data, but the aim of a DPM is to imitate the process underlying human similarity perception. Therefore, we need to perform experiments in an inherently human domain. As already mentioned, the selected domain is the detection of pedestrians in images.

(a) Positive Image

(b) Negative Image

Figure 3.3: Examples From the Dataset

Because much research in this area has been conducted, many datasets do exist. We selected the INRIA dataset[5] with upright images of persons in everyday situations. The dataset is 970 MB large and contains thousands of images. Example images are shown in Figures 3.3a and 3.3b.

To keep computation time low, the INRIA dataset is sub-sampled. Training is performed with 140 positive and 160 negative samples. Testing is done with 50 positive and 50 negative images. Positive training images are restricted to a size close to $96 * 160$ pixels, while negative training image can have any size larger than this. The reason is that the training algorithm works with one correctly sized, randomly taken sub-image for negative training images.

Technically, test images could have any size as long as we work with correctly sized sub-images, for example created by a simple iterative algorithm. To save computation time, this was not done during the experiments. Therefore, positive test images also have a size close to $96 * 160$ pixels. Again, because of the random sub-images, the size of the negative test images does not matter.

During SVM training, the number of allowed iterations without progress was restricted to 3000.

We perform a manual classification into thematic and taxonomic measures. If there is a contrast (i.e. $x - y$, $\frac{x}{y}$, $\frac{a}{\cdot}$, $\frac{\cdot}{-a}$, $\frac{\cdot}{b}$ or $\frac{\cdot}{c}$), then a measure is thematic and belongs on the right-hand side of Equation 1.1. Otherwise, it is taxonomic and belongs on the left-hand side.

The importance of taxonomic thinking was set to $\alpha = \frac{1}{2}$ during all experiments. This means we simulate a person that values taxonomic thinking as much as thematic thinking. The fixed importance factor does not pose a large restriction, because our image classification task can be performed correctly by virtually all adults, regardless of their preference for taxonomic or thematic thinking. This makes it very likely that our α simulates a large subset of all humans. Furthermore, α can easily be changed in our implementation.

With all implemented quantitative and predicate-based measures classified into taxonomic and thematic, we run pedestrian detection with the described dataset for:

- All combinations of quantitative measure/quantitative measure/generalization

[5] `http://pascal.inrialpes.fr/data/human` last accessed 2015-02-24

function.

- All combinations of predicate-based measure/predicate-based measure/generalization function.
- The most promising combinations of predicate-based measure/quantitative measure/generalization function.
- The most promising combinations of quantitative measure/predicate-based measure/generalization function.

With "the most promising combinations" we mean that only predicate-based and quantitative measures were used, that were part of a purely predicate-based or purely quantitative DPM that performed as good as the linear kernel. The reason for this restriction is to make search space smaller to decrease the already large runtime of the experiments.

To be able to compare our DPMs to the current state-of-the-art, we also ran pedestrian detection with the described dataset for the linear, polynomial, sigmoid and radial kernels. Additionally, the thematic and taxonomic parts of each DPM were used on their own to classify the test data set.

Furthermore, we carried out five experiments with selected DPMs and a larger dataset. This dataset contained 2379 positive and 1231 negative training samples. Testing in this case was done with 900 samples.

Some experiments did not terminate while training the support vector machine. This models have been omitted from the result discussion.

In this chapter, we took a closer look at the implementation details. The implementation consists of two main parts: the pedestrian detection part and the SVM part. While the former "glues" everything together, the latter can be used on its own to carry out experiments in other domains and to solve machine learning tasks other than pedestrian detection.

We saw how to work with the implementation and took a closer look at the experimental setup. We will discuss the outcome of those experiments in the next chapter. At this point, let us think back to our goals from the introductory chapter. We wanted to provide an implementation of the DPM idea that could perform binary classification and was adoptable for other tasks.

Clearly, this task was fulfilled. The question how a DPM can be integrated into machine learning was answered. Because of the very useful concept of kernel functions, it was more of a technical than a logical question.

CHAPTER 4

Results

We will now deal with the second aim of this work and asses the performance of DPMs for pedestrian detection. Using the experimental setup from before, we tested most of the DPMs that can be formulated with Equation 1.1.

4.1 Viability of the Dual Process

Using a Dual Process Model adds logical complexity to any machine learning task. Furthermore, depending on the actual measures used, it worsens time complexity. Is the overhead worth it?

We compare the classification performance of each DPM with its single process models, i.e. the taxonomic part $m_{taxonomic}$ with the thematic part $g(m_{thematic})$. Note that both parts measure similarity, because otherwise it would not be possible to use them as kernel functions.

The aggregated results of this comparison are shown in Table 4.1. It was created by measuring the classification performance of each possible DPM and comparing it to the classification performances of the two single process models that belong to it.

Aspect	**% of Cases**
Taxonomic Thinking Better Than Thematic Thinking	71,12
Thematic Thinking Better Than Taxonomic Thinking	19,03
Single Process Model Better Than or Equal DPM (Not Viable)	73,77
Single Process Model Worse Than DPM (Viable)	26,23

Table 4.1: Viability of DPMs

We can see in the first two lines that taxonomic thinking on its own often performed better than the thematic thinking on its own. This can be seen as a clue that taxonomic thinking has a larger impact on image detection performance than thematic thinking.

However, it is equally likely that this difference is caused by the way we combined taxonomic and thematic thinking in our model or by the algorithm selection for feature vector extraction.

The last two lines are more important. They tell us that DPMs are not necessarily better than single process models. In other words: Not every DPM created with Equation 1.1 makes sense. In about 74% of the search space, the classification performance has nothing to gain from the use of a DPM with fixed importance factor $\alpha = \frac{1}{2}$ and might even decrease. Therefore, when formulating a DPM, it is essential to verify that it improves performance for the task at hand.

Let us call this performance improvement viability. For the rest of the result discussion, we will exclude all DPMs that are not viable. It should be mentioned that using the Euclidean distance (a specific case of the Minkowski distance) often resulted in strong classification performance. However, because of our viability constraint, this measure does not appear often in the following results.

4.2 Comparison to Existing Models

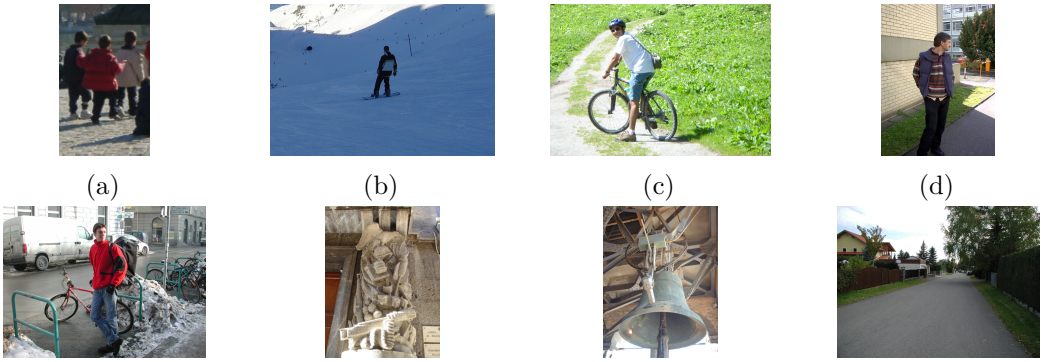
Are DPMs better for image classification than the current state of the art? To clear that, we compared them to linear, radial, polynomial and sigmoid kernels. Table 4.2 shows a summary of the classification performance for our pedestrian detection task.

Kernel	**% Correct**
Sigmoid	50
Radial	50
Linear	92
Polynomial	92
Baroni+Shepard(Normalization)	96
Histogram+Shepard(Minkowski)	95
Tanimoto Index+Boxing(Complement of Hamming Distance)	94
Russel & Rao-Minkowski	94
...	...

Table 4.2: Comparison to the State of the Art

A striking characteristic of our comparison is that the values 50% and 92% seem to appear more often than they should. This has a compelling reason: the dataset, which contains an equal amount of positive and negative images to classify. Regularly, suboptimal DPM kernels have a tendency to classify all images as positives or negatives. In this case, the classification score is always obviously biased, e.g. $y = 300$ or $y = -500$. With our experimental setup, if a kernel only produces one classification result, the correct classification rate will always be 50%.

Another property of the dataset is that it contains some images that are fairly difficult to classify. Classification errors agglomerate for the same images. As a result, the



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.1: Difficult Test Images

linear and polynomial kernel make their classification errors for the same images. We can see them in Figure 4.1. Still, the classification scores of the linear kernel and the polynomial kernel are different. Therefore, it is not very plausible that this is a bug in the implementation or in the experimental setup.

The difficult images are representative for situations that are hard for most image detection algorithms: bad lighting (Figures 4.1a, 4.1b, 4.1d and 4.1h) and entangled shapes (Figures 4.1c, 4.1e, 4.1f and 4.1g).

Note that the principal goal of the experimental setup and the implementation was not to come up with image classification that is as optimized as possible. It would have been counter-productive to optimize the feature vectors until all test images could be classified correctly.

We need some errors that are left to examine the effect of DPMs. Therefore, all results have to be seen relative to the existing kernels and not to state of the art image classification algorithms. In particular, we did not perform multiple training rounds. That is, we did not add the images from Figure 4.1 to our training data after we learned that they are hard to classify.

How many correct classifications do we expect from a kernel? The error tolerance in real world tasks is often low. Therefore, we expect a correct classification rate of at least 90% from a well-performing kernel. This limit has been chosen somewhat arbitrarily, but some limit is definitely needed, because it does not make sense to examine kernels that barely work.

Experiments showed that sigmoid and radial kernels performed poorly, while the widely-used linear and polynomial kernels performed well. Many tested DPMs made less than 90% correct test data classifications. However, about 9% of all DPMs performed that well or better. Refer to Section 4.6 for more information about these DPMs.

The precision/recall-curve of all well-performing models from Table 4.2 is shown in Figure 4.2. We did not find remarkable behavior of DPM kernels in the high precision or high recall areas. However, a more thorough quantitative analysis of the precision/recall-curves was out of scope for this work.

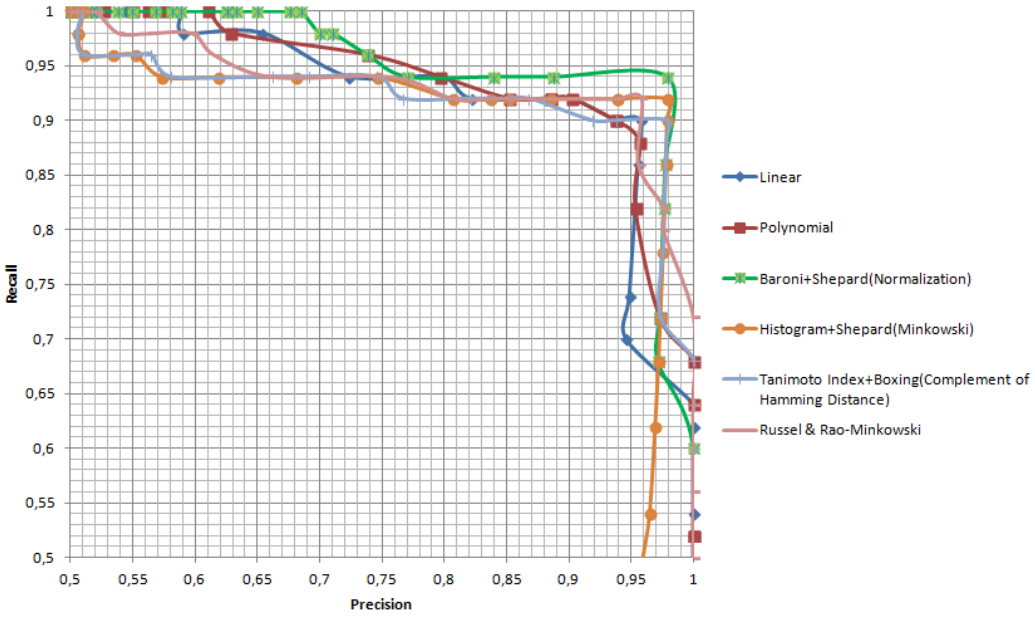


Figure 4.2: Precision/Recall-Curve for Conventional Kernels and DPM Kernels

4.3 Generalization

All DPMs were tested with different generalization functions (refer to Section 2.3 and Chapter 5 in the appendix for details): Boxing, Gaussian, Shepard and None.

We group all DPMs with a classification performance of at least 90% by their generalization function. The results can be seen in Table 4.3.

Generalization Function	**% of High-performing DPMs**
Shepard	51,0
None	23,5
Gaussian	23,5
Boxing	2,0

Table 4.3: Percentage of High-performing DPMs per Generalization Function

The Shepard generalization function was most often part of high-performing DPMs. Surprisingly, not using any generalization function proved as successful as using the Gaussian generalization function. The Boxing function did not work well, but we found that its performance increases if additional iterations were allowed during SVM training.

The data show that if a combination of taxonomic and thematic measure is successful with one generalization function, it tends to be successful with other generalization functions, too. The probability of arriving at a high-performing DPM is highest when using the Shepard generalization function. However, good classification performances

Taxonomic	Generalization	Thematic	%
Sorgenfrei	Shepard	Batagelj & Bren	90
Hawkins & Dotson	Shepard	Variance Dissimilarity	90
Baroni-Urbani & Buser	Shepard	Baulieu Variant 2	90
Coeff. of Arith. Means	Shepard	Baulieu Variant 2	90
Proportion of Overlap	Shepard	Baulieu Variant 2	90

Table 4.4: Classification Performance of Predicate-based DPMs

could be obtained with most generalization functions, as long as fitting thematic and taxonomic measures were chosen.

Based on the experiments, selecting the taxonomic and thematic measure seems to have a much larger impact on the classification performance of DPMs. In the opinion of the author, generalization will only begin to have a larger impact, if we reach a level of model complexity where the generalization function is constructed dynamically from observed generalization gradients.

4.4 Quantitative and Predicate-based Measures

As already discussed, quantitative measures operate on real-valued feature vectors, while predicate-based measures operate on 0/1 values. To keep the number of experiments manageable, we first had to test all purely quantitative and all purely predicate-based DPMs. Only the most promising measures of these experiments where tested in combination. This approach is inspired by genetic algorithms.

The experiments indicated that if quantitative measures are used exclusively, their performance is slightly better than the exclusive use of predicate-based measures. Table 4.4 states the classification performance of the best DPMs that use only predicate-based measures.

Like before, 90% seems to appear more often than it should. Again, this is explained by the difficult test images that lead to the same errors for all shown DPMs. Hence, our predicate-based feature vector extraction is not discriminative enough.

Table 4.5 shows the best DPMs that use only quantitative measures. Their correct classification rate is always a little bit higher than the rate of their predicate-based counterparts.

Figure 4.3 compares purely quantitative to purely predicate-based DPMs by plotting their precision/recall-curves against each other. The first two measures in the legend, blue and red, are predicate-based and the last two, green and violet, are quantitative DPMs. We see that quantitative measures seem to have an advantage in the high recall area, that vanishes quickly if more precision is required. Further analysis of this finding was out of scope of this work.

Until now, our DPMs used either quantitative or predicate-based measures only - but these two types of measures can be mixed. Table 4.6 summarizes the classification

Taxonomic	Generalization	Thematic	%
Histogram Intersection	Shepard	Minkowski Distance, Meehl Index	95
Histogram Intersection	Shepard	Kullback & Leibler, Jeffrey Divergence	95
Histogram Intersection	Shepard	Exponential Divergence, Normalization	95
Histogram Intersection	Shepard	Kagan Divergence, Mahalanobis Distance	95
Tanimoto Index	Shepard	Minkowski Distance	94
Modified Dot Product	Gaussian	Minkowski Distance, Mahalanobis Distance	93
Modified Dot Product	Shepard	Mahalanobis Distance	93
Cosine Measure	Gaussian	Minkowski Distance, Mahalanobis Distance	93
Cosine Measure	Shepard	Mahalanobis Distance	93
Tanimoto Index	Shepard	Normalization, Mahalanobis Distance	92

Table 4.5: Classification Performance of Quantitative DPMs

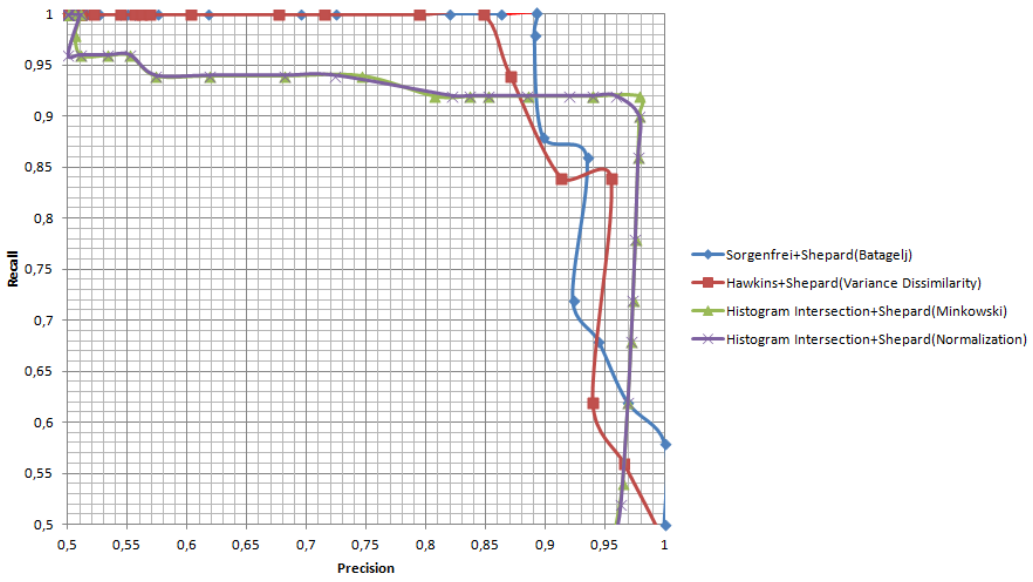


Figure 4.3: Precision/Recall-Curve for Selected Quantitative and Predicate-based Measures

performance of the best mixed DPMs.

Note that some mixed DPMs appear in Table 4.6 that were not part of the best purely quantitative DPMs or purely predicate-based DPMs. The reason for this is that DPMs that performed well, but were not viable, were also permitted to take part in the mixed test round. However, all of the mixed DPMs are still required to be viable.

We can see that mixed DPMs work as well as quantitative or predicate-based DPMs. This is an encouraging result, because it allows us to select our DPM parts based on the feature vector type at hand.

Against intuition, mixing quantitative measures with predicate-based measures (that performed slightly weaker in general), still often lead to improved classification performance. This is further empirical evidence in support of DPMs.

In the beginning of this section, we hinted at quantitative DPMs being slightly superior to their predicate-based counterparts. In the light of the performance of mixed DPMs, this does not hold. The observed performance difference is likely to be caused by the algorithms used for feature vector extraction.

4.5 Statistical Significance

Because of the long runtime of each experiment, it was not possible to test each DPM many times with different subsamples and provide confidence intervals for the classification results. It is plausible that in other experiments, a few of the well-performing measures disappear or new ones are found. What we can show is that DPMs work in principle and our results are not just caused by the degrees of freedom our large search space provides.

Assume for a moment that DPMs do not work at all and their classification is utterly random. In a binary classification problem, the chances of guessing the class label correctly are 50%, which is quite high. Our generic model for constructing DPMs allows for many different concrete DPMs. That means we would have many chances to guess a good test data classification.

One argument against our findings in this chapter being random are the good classification results of the best mixed DPMs. We restricted the search space for this experiments to DPM parts that already showed a good classification performance in purely quantitative or purely predicate-based DPMs. Still, we got good classification results again.

As further evidence, we ran two of the best DPMs again with a large sample. The classification results are shown in Table 4.7.

We can see that the classification performance for a much larger dataset with nine times as many classifications remained stable. This indicates that our results are not fragile in the sense that for every dataset, different DPMs show up with high classification performances. However, it has to be noted that much stronger propositions would have been possible, if the size and total runtime of the experimental setup had permitted the calculation of confidence intervals.

Taxonomic	**Generalization**	**Thematic**	**%**
Baroni-Urbani & Buser	Shepard	Normalization	96
Sorgenfrei	Shepard	Normalization	95
Coeff. of Arith. Means	Shepard	Normalization	95
Russel & Rao	Shepard	Exponential Div.	94
Russel & Rao	None	Minkowski Dist.	94
Russel & Rao	None	Exponential Div.	94
Tanimoto Index	Boxing, Gaussian	Compl. of Hamming Dist.	94
Tanimoto Index	Shepard	Compl. of Hamming Dist.	94
Correlation Coefficient	Shepard, None	Baulieu Var. 2	94
Correlation Coefficient	Shepard, None	Batagelj & Bren	94
Correlation Coefficient	Shepard	Variance Dis.	94
Histogram Intersection	None	Hamming Dist.	94
Russel & Rao, Sorgenfrei	Gaussian	Normalization	93
Baroni-Urbani & Buser	Gaussian	Normalization	93
Coeff. of Arith. Means	Gaussian	Normalization	93
Proportion of Overlap	Gaussian, Shepard	Normalization	93
Sorgenfrei	None	Normalization	93
Proportion of Overlap	None	Exponential Div.	93
Cosine Measure	Boxing	Compl. of Hamming Dist.	93
Mod. Dot Prod.	Gaussian	Variance Dis., Batagelj & Bren	93
Cosine Measure	Gaussian	Baulieu Var. 2	93
Cosine Measure	Gaussian	Compl. of Hamming Dist.	93
Cosine Measure	Shepard, None	Variance Dis., Baulieu Var. 2	93
Cosine Measure	Shepard, None	Batagelj & Bren	93
Cosine Measure	Shepard, None	Compl. of Hamming Dist.	93
Tanimoto Index	Gaussian	Baulieu Var. 2	93
Tanimoto Index	Shepard, None	Variance Dis., Baulieu Var. 2	93
Tanimoto Index	Shepard, None	Batagelj & Bren	93
Russel & Rao	Gaussian	Exponential Div.	92
Hawkins & Dotson	Shepard, None	Exponential Div.	92
Baroni-Urbani & Buser	Shepard	Exponential Div.	92
Coeff. of Arith. Means	Shepard, None	Exponential Div.	92
Proportion of Overlap	Shepard	Exponential Div.	92
Mod. Dot Prod.	Gaussian	Baulieu Var. 2	92
Mod. Dot Prod.	Shepard, None	Variance Dis.	92
Hawkins & Dotson	Gaussian	Normalization	91
Sorgenfrei	Shepard, None	Exponential Div.	91
Hawkins & Dotson	Gaussian	Exponential Div.	90
Hawkins & Dotson	Shepard	Normalization	90
Coeff. of Arith. Means	None	Normalization	90

Table 4.6: Classification Performance of Mixed DPMs

Taxonomic	Generalization	Thematic	%
Baroni-Urbani & Buser	Shepard	Normalization	95,4
Histogram Intersection	Shepard	Minkowski Distance	94,6

Table 4.7: Classification Performance With a Larger Dataset

4.6 Constructing DPMs

Until now, we always stated concrete combinations of taxonomic measure, thematic measure and generalization function. Just using these combinations seems too narrow-minded. The data show that many measures always work well, but are often dragged down by combining them with measures with poor classification performance.

Therefore, we provide instructions for building well-performing DPMs by listing all measures and generalization functions that were part of at least one viable model with a classification performance of at least 90%. Our options for well-performing DPMs are shown in Tables 4.8, 4.9, 4.10, 4.11 and 4.12.

For clarity, we split the options into quantitative and predicate-based DPMs, but they can be mixed at will. They are intended to be plugged into Equation 1.1. Refer to Chapter 5 in the appendix for exact definitions.

This concludes the results chapter. Unsurprisingly, using Equation 1.1 does not work with all measures. Quite to the contrary, most measures barely work at all when used in a DPM.

One possible reason is that the classification into taxonomic and thematic measures might not be as simple as described in Section 3.5. Our experimental setup labeled every given measure either as taxonomic or as thematic, but maybe measures exist that are neither taxonomic nor thematic. Such measures would show in our results with very poor performance. This would explain the myriad of DPMs with poor performances that we observed during the experiments.

An alternative explanation has to do with the structure of Equation 1.1 itself. It can be described as adding taxonomic and thematic similarity to arrive at a total similarity score. It is not a given that both taxonomic and thematic similarity have the same magnitude for all existing measures. As an example for what could go wrong, assume that $s_{dpm} = 100 + 3$. A similarity of three could mean a huge similarity for the taxonomic measure, while a similarity of 100 only means "somewhat similar" for the thematic measure.

On the plus side, we found some concrete DPMs that perform well. From this, we can conclude that psychological research in the area of thematic and taxonomic thinking has useful applications in machine learning.

The practitioner, when faced with a specific machine learning task, can just select two measures from our tables and check whether the DPM kernel constructed like this works better than the linear or polynomial kernel. The decision whether to use predicate-based or quantitative measures is only a technical one and depends just on the format of the input data.

The question about the best-performing DPMs has not been answered fully. We know that if following our building instructions, the performance of the linear kernel and polynomial kernels can usually be matched. With our data set and feature extraction, we are in the realm of about 90% correct classifications for both conventional kernels and DPM kernels. How DPMs will compare to the state of the art of other machine learning tasks, or even other feature extraction algorithms, cannot be said yet and is left as a task for future research.

g
Boxing
Gaussian
Shepard
None

Table 4.8: Generalization Functions

$m_{taxonomic}$
Histogram Intersection
Modified Dot Product
Tanimoto Index
Correlation Coefficient
Cosine Measure

Table 4.9: Quantitative Taxonomic Measures

$m_{thematic}$
Meehl Index
Minkowski Distance
Exponential Divergence
Kagan Divergence
Jeffrey Divergence
Kullback Leibler
Mahalanobis Distance
Normalization

Table 4.10: Quantitative Thematic Measures

$m_{taxonomic}$
Hawkins Dotson
Sorgenfrei
Russel & Rao
Baroni Urbani Buser
Proportion of Overlap
Coefficient of Arithmetic Means

Table 4.11: Predicate-based Taxonomic Measures

$m_{thematic}$
Variance Dissimilarity
Batagelj Bren
Baulieu Variant 2
Hamming Distance
Complement of Hamming Distance

Table 4.12: Predicate-based Thematic Measures

CHAPTER 5

Conclusion and Future Work

In this thesis, we briefly introduced existing models of similarity like geometric models, featural models and transformational models. DPMs are meta models for measuring similarity. They combine taxonomic and thematic thinking.

Taxonomic thinking tries to identify common features between two objects. The more common features, the larger the similarity. Taxonomic thinking is best represented by similarity measures, i.e. measures that become larger with the similarity of two stimuli.

Thematic thinking tries to find a common theme that connects two objects. This theme is then used for comparison. Thematic thinking is best represented by distance measures, i.e. measures that become smaller the larger the similarity of two objects gets.

Using a generalization function, distance can be converted into similarity. In Equation 1.1, we used this to formulate a simple DPM that measures similarity.

Psychological research suggests that a combination of both taxonomic and thematic thinking is needed to model human similarity assessment adequately. Similarity is often an important part of machine learning algorithms. However, one of the two ways of thinking is often neglected - especially in models formulated by computer scientists.

We implemented pedestrian detection in images to be able to test DPMs in a real world task. Input images have to be converted into feature vectors. We used a combination of three algorithms for this task: Histogram of Oriented Gradients, Scalable Color Descriptor and Edge Histogram Descriptor. The feature vectors were used to train a SVM, which is a binary classification algorithm. Eventually, the SVM classifies test images into “contains a pedestrian” and “contains no pedestrian”.

SVMs use a kernel function to measure similarity. Some kernel functions also map input data to a higher-dimensional space to achieve better data separation. We provided DPM kernel functions for the widely-used SVM^{light} library. The implementation lets us choose from 5 generalization functions and 94 measures. It is freely available, because the main goal of this work was to provide such an implementation to foster further work in the area of DPMs.

We tested our DPMs with a subsample of a standard dataset and got some interesting results. Any DPM is a combination of two measures. Obviously, if using just one measure performs as well or better than using two measures, we do not deal with a viable DPM. Only 14% of our DPMs were found to be viable. The takeaway point here is that when using a DPM, one should always check whether the classification performance does not just come from one measure alone.

DPMs can be formulated with quantitative measures (i.e. real values), predicate-based measures (i.e. countable or 0/1 values) and with a mix of both types of measures. We did not find conclusive evidence that a certain measure type (e.g. measuring taxonomic and thematic thinking with quantitative measures) works better than any other type.

We discovered DPMs that performed as good as or better than the existing linear and polynomial kernels. Let us, for example, combine the Russel and Rao similarity measure with the Exponential Divergence difference measure using Shepard's generalization function. This is shown in Equation 5.1.

$$m_{dpm} = \frac{a}{a+b+c+d} + e^{-|\sum_i x_i log(\frac{x_i}{y_i})^2|} \tag{5.1}$$

Russel and Rao's measure calculates the percentage of shared features. This is a simple improvement over just counting the co-occurrences. The Exponential Divergence measure represents an entirely different way of thinking about similarity by measuring information gain.

However, it has to be mentioned that this is not a mandatory proof that DPMs outperform the current state of the art. With the algorithms we used, meaningful feature vectors contain thousands of elements per image. This makes SVM training (and even classification) slow for conventional kernels as well as DPM kernels. In addition, the search space is large, because many concrete DPMs can be formulated with Equation 1.1.

Conclusive evidence would have to carry out statistical testing to be able to state significance levels of classification performances. For this, every single specific DPM has to be tested many times. To make this possible, runtime has to be improved.

Future work could either use algorithms that yield good classification results with much smaller feature vectors or focus only on a few possible DPMs. To support this, we provided a construction kit for well-performing DPMs.

Another interesting direction for further research are DPMs in other domains, for example audio and text retrieval or non-multimedia problems like recommender systems and computational finance. Because DPMs are kernel functions, they can be readily used with algorithms other than SVMs like Gaussian processes, ridge regression, spectral clustering and many more.

In this work, a constant importance factor controlled the relative weight of taxonomic and thematic thinking. We assumed it can be found with psychological tests. It is very likely that the importance factor is situation-dependent rather than constant. Making the importance factor a function of the features would provide another promising direction of research.

Hopefully, this work provides a good basis for further work on DPMs. Improved similarity measurement processes would not only lead to large improvements on the

application-side, but also bring us closer to the noble goal of understanding similarity as a concept.

Appendix: Formulas

Below, x, y are the feature vectors with known means μ_x, μ_y and known standard deviations σ_x, σ_y. K is the size of both feature vectors. a, b, c, d describe the feature vectors as explained in Section 2.4.2.

Quantitative Measures

$$HistogramIntersection = \sum_i min(x_i, y_i) \tag{2}$$

$$ModifiedDotProduct = \frac{\sum_i x_i y_i}{K} \tag{3}$$

$$TanimotoIndex = \frac{\sum_i x_i y_i}{\sum_i x_i^2 + \sum_i y_i^2 - \sum_i x_i y_i} \tag{4}$$

$$CorrelationCoefficient = \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_i (x_i - \mu_x)^2} \sqrt{\sum_i (y_i - \mu_y)^2}} \tag{5}$$

$$CosineMeasure = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2}} \tag{6}$$

$$MeehlIndex = \sum_i^{K-2} ((x_i - x_{i+1}) - (y_i - y_{i+1}))^2 \tag{7}$$

We set $a_2 = a_1 = 2$ in Equation 8, i.e. we used the Euclidean Distance for our experiments.

$$MinkowskiDistance = \sqrt[a_2]{\frac{|x_i - y_i|^{a_1}}{K}} \tag{8}$$

$$ExponentialDivergence = \sum_i x_i log(\frac{x_i}{y_i})^2 \tag{9}$$

$$KaganDivergence = \frac{1}{2}\sum_i \frac{(x_i - y_i)^2}{x_i} \tag{10}$$

$$JeffreyDivergence = \sum_i (x_i - y_i) log(\frac{x_i}{y_i}) \tag{11}$$

$$KullbackLeibler = \sum_i x_i log(\frac{x_i}{y_i}) \tag{12}$$

$$MahalanobisDistance = \sum_i \sum_j (x_i - y_i)(x_j - y_j) \tag{13}$$

Equation 14 is the most successful attempt of the author of this work. It is a very simple normalization that has certainly be used to measure similarity before.

$$Normalization = |\frac{\mu_x}{\sigma_x} - \frac{\mu_y}{\sigma_y}| \tag{14}$$

Predicate-based Measures

$$HawkinsDotson = \frac{1}{2}(\frac{a}{a+b+c} + \frac{d}{b+c+d}) \tag{15}$$

$$Sorgenfrei = \frac{a^2}{(a+b)(a+c)} \tag{16}$$

$$RusselRao = \frac{a}{a+b+c+d} \tag{17}$$

$$BaroniUrbaniBuser = \frac{\sqrt{ad} + a - b - c}{\sqrt{ad} + a + b + c} \tag{18}$$

$$ProportionOfOverlap = \frac{ad - bc}{K(1 - \frac{a}{(a+b)(a+c)})(2a + b + c - \frac{(a+b)(a+c)}{K})} \tag{19}$$

$$CoefficientOfArithmeticMeans = \frac{2(ad - bc)}{K(2a + b + c)} \tag{20}$$

$$VarianceDissimilarity = \frac{b + c}{4K} \tag{21}$$

$$BatageljBren = \frac{bc}{ad} \tag{22}$$

$$Baulieu2 = \frac{K(b+c) - (b-c)^2}{K^2} \tag{23}$$

$$HammingDistance = b + c \tag{24}$$

$$ComplementOfHammingDistance = a + d \tag{25}$$

Generalization Functions

We set $\rho = 0.6$ in Equation 26 for our experiments.

$$Boxing = \begin{cases} 1, & \text{if } -\rho \leq distance \leq \rho \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

$$Gaussian = e^{-distance^2} \tag{27}$$

$$Shepard = e^{-|distance|} \tag{28}$$

$$None = -distance \tag{29}$$

Bibliography

[Att54] Fred Attneave. Some Informational Aspects of Visual Perception. *Psychological Review*, 61(3):183, 1954.

[BcGOU09] Muhammet Baştan, Hayati Çam, Uğur Güdükbay, and Özgür Ulusoy. BilVideo-7: An MPEG-7-Compatible Video Indexing and Retrieval System. *IEEE MultiMedia*, 17(3):62–73, 2009.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*, volume 1. Springer New York, 2006.

[DT05] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[DWSP12] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian Detection: An Evaluation of the State of the Art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.

[EGJ11] Zachary Estes, Sabrina Golonka, and Lara L. Jones. Thematic Thinking: The Apprehension and Consequences of Thematic Relations. *Psychology of Learning and Motivation-Advances in Research and Theory*, 54:249, 2011.

[Eid12] Horst Eidenberger. *Handbook of Multimedia Information Retrieval*. BoD–Books on Demand, 2012.

[GS05] Robert L. Goldstone and Ji Yun Son. *Similarity*. Cambridge University Press, 2005.

[Ima77] Shiro Imai. Pattern Similarity and Cognitive Transformations. *Acta Psychologica*, 41(6):433–447, 1977.

[Joa98] Thorsten Joachims. Making Large-Scale SVM Learning Practical. LS8-Report 24, Universität Dortmund, LS VIII-Report, 1998.

[OCK01] Jens-Rainer Ohm, Leszek Cieplinski, and Heon Jun Kim. The MPEG-7 Color Descriptors. *Introduction to MPEG-7: Multimedia Content Description Interface, Wiley*, 2001.

[OMCS04] Cheng Soon Ong, Xavier Mary, Stéphane Canu, and Alexander J. Smola. Learning With Non-positive Kernels. In *Proceedings of the Twenty-first International Conference on Machine Learning*, page 81. ACM, 2004.

[PBKE12] Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. Real-Time Computer Vision With OpenCV. *Communications of the ACM*, 55(6):61–69, 2012.

[PG79] Peter Podgorny and W. R. Garner. Reaction Time as a Measure of Inter- and Intraobject Visual Similarity: Letters of the Alphabet. *Perception & Psychophysics*, 26(1):37–52, 1979.

[She87] Roger N. Shepard. Toward a Universal Law of Generalization for Psychological Science. *Science*, 237(4820):1317–1323, 1987.

[Sik01] Thomas Sikora. The MPEG-7 Visual Standard for Content Description-An Overview. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):696–702, 2001.

[TG82] Amos Tversky and Itamar Gati. Similarity, Separability, and the Triangle Inequality. *Psychological Review*, 89(2):123, 1982.

[TG01] Joshua B. Tenenbaum and Thomas L. Griffiths. Generalization, Similarity, and Bayesian Inference. *Behavioral and Brain Sciences*, 24(04):629–640, 2001.

[Tve77] Amos Tversky. Features of Similarity. *Psychological Review*, 84:327–352, 1977.

[Vap00] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

[VTS04] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A Primer on Kernel Methods. *Kernel Methods in Computational Biology*, pages 35–70, 2004.

[WB99] Edward J. Wisniewski and Miriam Bassok. What Makes a Man Similar to a Tie? Stimulus Compatibility With Comparison and Integration. *Cognitive Psychology*, 39(3):208–238, 1999.

[WHK95] Edward A. Wasserman, Jacob A. Hugart, and Kim Kirkpatrick-Steger. Pigeons Show Same-different Conceptualization After Training With Complex Visual Stimuli. *Journal of Experimental Psychology: Animal Behavior Processes*, 21(3):248, 1995.