

# Kernel Methods for Image Processing

Dan George Bucatanschi

Project Advisor: R. Matthew Kretchmar  
Department of Mathematics & Computer Science

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the work, and its date appear, and notice is given that copying is by permission of the author. To copy otherwise, to republish, to post on a server, or to redistribute to lists, requires prior specific permission of the author and/or a fee. (Opinions expressed by the author do not necessarily reflect the official policy of Denison University.)

Copyright, Dan George Bucatanschi, 2006

## **Kernel Methods for Image Processing**

Dan George Bucatanschi

Department of Mathematics & Computer Science

Advisor: R. Matthew Kretchmar

We discuss the possibility for a computer to identify people and human features, such as long hair and smiles, from photographs of human faces. These problems fall into the field of supervised learning. We begin with a discussion of the broader problem of classification and various ways in which it is traditionally solved. We introduce the kernel and its use in solving problems. We then present various algorithms and the methods they employ together with kernels to solve the supervised learning tasks.

We finish our discussion by presenting the results and conclusions of solving three tasks: identifying a person out of photographs of several people, detecting photographs of people with long hair, and finding photographs of smiling people.



# Table of Contents

<b>Chapter 1: Introduction</b> .....	<b>1</b>
1.1 Overview .....	1
1.2 What Is Artificial Intelligence? .....	2
1.3 Supervised Learning .....	3
1.3.1 Perceptrons .....	4
1.3.2 Neural Networks .....	5
1.3.3 Kernel Methods .....	7
1.4 Problem Details and Technical Notations .....	10
1.5 Previous Work .....	10
1.6 Paper Outline .....	11
<b>Chapter 2: Kernels</b> .....	<b>13</b>
2.1 Remapping Input Space to Feature Space .....	13
2.2 Definition of a Kernel .....	15
2.3 The Direct Method .....	15
2.4 The Kernel Trick .....	16
2.5 The Kernel Matrix .....	17
2.6 Examples of Kernels .....	18
2.6.1 The Inner Product .....	18
2.6.2 Polynomials of Inner Products .....	19
2.6.3 The Gaussian Kernel .....	20
<b>Chapter 3: Algorithms</b> .....	<b>23</b>
3.1 Role of Kernels .....	23
3.2 Datasets .....	24
3.3 Distance Algorithms .....	25
3.3.1 Naïve Novelty Detector .....	25
3.3.2 Parzan Window Classifier .....	26
3.3.3 Fisher Linear Discriminant Classifier .....	27
3.4 Support Vector Machines .....	28
3.4.1 SVM Novelty Detector .....	29
3.4.2 SVM Maximum Margin Classifier .....	29
3.5 Principal Components Analysis .....	30
<b>Chapter 4: Our Data</b> .....	<b>33</b>
4.1 Human Face Database .....	33
4.2 Ratings .....	34
<b>Chapter 5: Face Recognition</b> .....	<b>37</b>
<b>Chapter 6: Feature Identification—Hair Style</b> .....	<b>59</b>
<b>Chapter 7: Feature Identification—Smiling</b> .....	<b>85</b>
<b>Chapter 8: Conclusions</b> .....	<b>101</b>
<b>Acknowledgments</b> .....	<b>103</b>
<b>Bibliography</b> .....	<b>105</b>
<b>Appendix A: Images</b> .....	<b>107</b>



# Chapter 1: Introduction

## **1.1 Overview**

Since the beginnings of Computer Science, researchers have been trying to develop ways through which a machine can simulate certain human capabilities. The recent advances in Artificial Intelligence (AI) coupled with the ever increasing speed of computers and access to cheaper technology have made tasks such as computer speech recognition, computer human face detection and identification, and even computer aircraft piloting a possible reality, not an unimaginable future. Computer scientists, psychologists and other researchers have been trying to understand and emulate the human mind in various potentially useful applications.

In this paper we discuss the possibility for a computer to detect people and human features, such as long hair and smiles, from photographs of human faces. We present the various techniques we use, particularly the kernel methods we employ, and the different algorithms with which we experiment. We start with an overview of the field of AI and its subdivisions.

## 1.2 What Is Artificial Intelligence?

Artificial Intelligence is a branch of Computer Science that tackles different problems in a human- or biological-intelligence inspired way, and which relies on several other branches of Computer Science as well as other fields ranging from Psychology to Biology and Statistics. More concretely, AI solves problems heuristically in such areas as planning, searching, information processing and controlling devices. AI includes genetic algorithms, natural language processing, planning, searching, machine learning and others. In this paper we focus on machine learning.

Machine learning is further subdivided into the following categories based on the type of information available:

- *Unsupervised learning*. In this case the computer finds patterns without any intervention or help from the outside [9]. Applications include clustering of data and principal components analysis (PCA);
- *Reinforcement learning*. In this case for every generated solution, the computer is given a score, with the ultimate goal of maximizing the score. The score reflects how closely the solution achieves the goal of the problem. Basically this trial and error approach involves several runs, during which the algorithm searches for a solution that will earn the highest score [18];
- *Supervised learning*. For this approach the computer learns from examples given by a knowledgeable “teacher.” At the end of the learning session, the goal is for the computer to return a correct solution on new, unseen data.

The algorithms in this paper primarily fall under the umbrella of supervised learning.



### 1.3 Supervised Learning

In supervised learning the data is represented as vectors. We usually have a set of vectors  $X = \{ x_i: i = 1..l \}$  representing the input to the problem and a set of vectors  $Y = \{ y_i: i = 1..l \}$  representing the associated desired output, where  $l$  is the number of vectors in the dataset. We generally represent the data in pairs such as  $(x_i, y_i)$ . For example, if we want to train a machine to distinguish between pictures of apples and oranges, then  $X$  is the set of pictures of apples and oranges (each picture is reformatted as a vector), and  $Y$  is the set of associated labels for each picture such as  $y_1 = 0, y_2 = 1, y_3 = 1, y_4 = 0, y_5 = 1$ , and so forth, where 0 represents ‘orange’ and 1 represents ‘apple’.

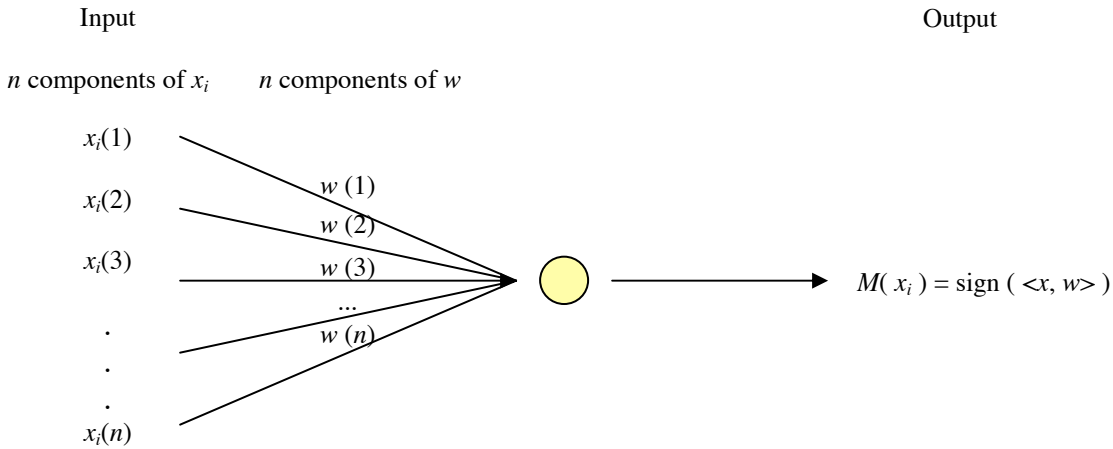
Given the training data, the goal of supervised learning is to learn a function  $M$  that maps each example input to the correct output, thus minimizing the sum  $\sum_i \|M(x_i) - y_i\|$ . Oftentimes the function  $M$  has a set of parameters  $\varphi$  as input so that we have  $M(\varphi, x_i) \mapsto y_i$ . In reality, most supervised learning algorithms work by a priori choosing a particular kind of function  $M$  and then automatically adjusting the parameters  $\varphi$  so that the mentioned sum is minimized.

*Classification* is a particular kind of supervised learning in which  $y_i \in \{c_1, \dots, c_p\}$  where each  $c_j$  is a category and there are  $p$  categories [5]. *Binary classification* is the case where  $p = 2$ . The “apples and oranges” learning task is a binary classification task. An example of a non-binary classification task is character recognition from handwritten text. In this situation  $p = 62$  (26 lower case letters, 26 upper case letters, and 10 digits).

Moreover, the data is represented as  $n$ -dimensional vectors. Thus, each training and testing point  $x \in \mathfrak{R}^n$  lives in an  $n$ -dimensional space.

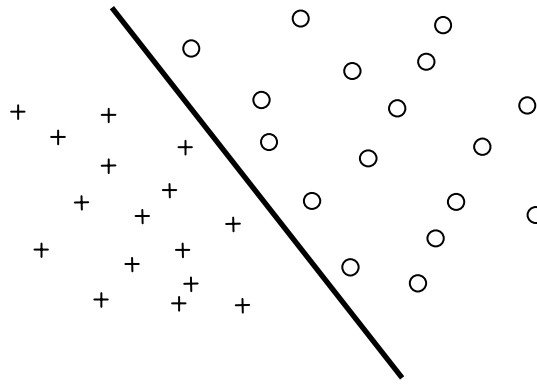
### 1.3.1 Perceptrons

In the beginnings of supervised learning, *perceptrons* were the fundamental learning machines [5, 6, 7]. They are binary classification machines that use a weight  $w = \varphi$  applied to the input to compute their output where  $w \in \mathfrak{R}^n$  (see Figure 1.1).



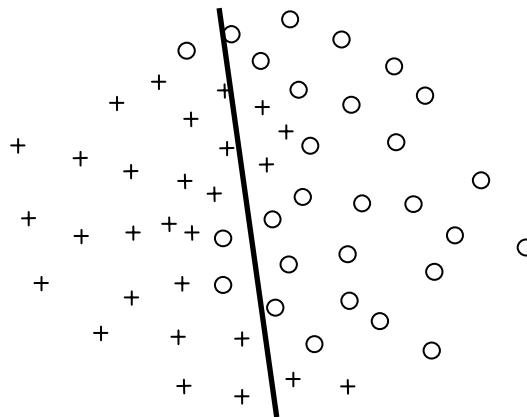
**Figure 1.1. Overview of a Perceptron. Note:  $x_i$  and  $w$  are vectors.**

The goal of the perceptron is to minimize the classification error  $\sum_i \|M(\varphi, x_i) - y_i\|$  for a particular problem. For example in Figure 1.2 we have a set of 2-dimensional data points  $x_i$  for which the desired output  $y_i \in \{+, 0\}$ . Visually, a perceptron would adjust the weight  $w$  to find a line that separates the two categories of points. Moreover, the perceptron has precise performance guarantees. This means that given a particular size of the training set, the perceptron is expected to classify a particular percentage correct output in the future, which can be computed just from the training data.



**Figure 1.2. Linearly Separable Data Set with Line Separating the Two Types of Data.**

However, the perceptron can learn only linearly separable data, i.e. the decision boundary is a straight line. For a data set distributed like in Figure 1.3, the perceptron's limitation would force it to misclassify several points.

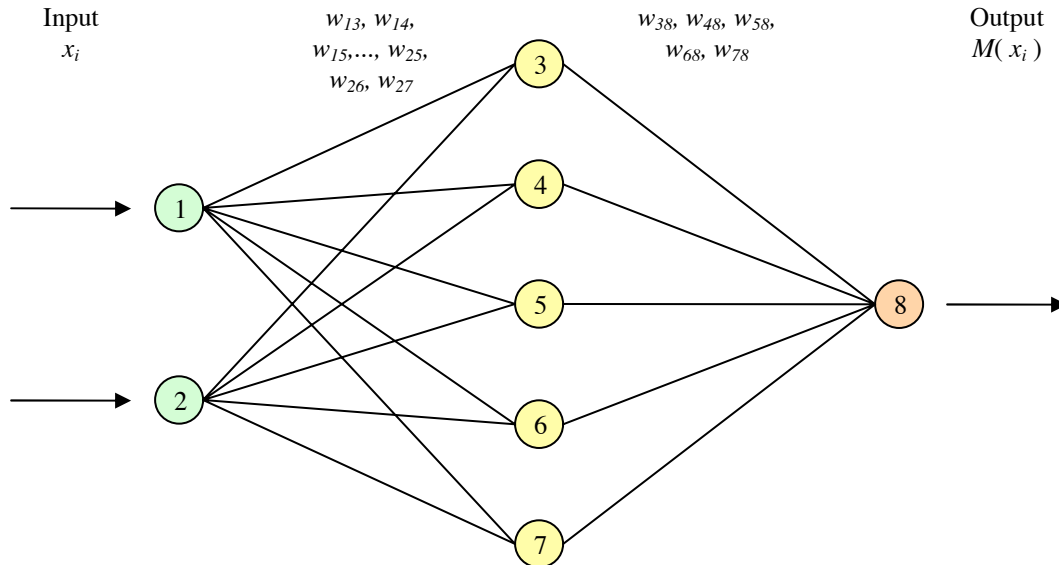


**Figure 1.3. Non-Linearly Separable Data Set with Line Separating the Two Types of Data. Note the misclassification of the points.**

### 1.3.2 Neural Networks

For some problems the perceptron is not adequate since many learning tasks need to separate non-linearly separable data points. In other words, we need an algorithm that is able to learn a non-linear function as seen in Figure 1.3. One such approach is the *neural network* [6, 7]. It is actually composed of several perceptrons, linked in a

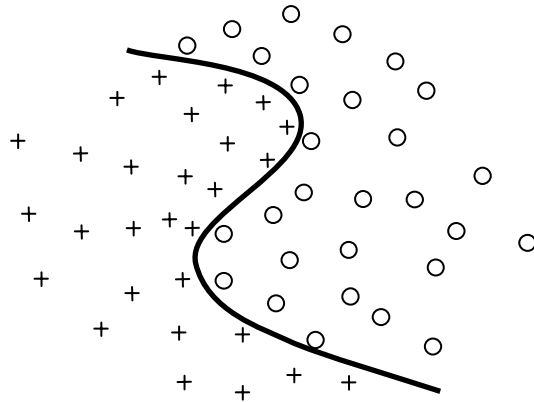
particular way such that the weights  $w = \varphi$  are applied to two or more layers of perceptrons (see Figure 1.4).



**Figure 1.4. Overview of a Neural Network.**

The weights of each link,  $w_{ij}$ , are learned similarly to the perceptron case, but usually require more training example to be trained optimally. In fact, the number of weights in a neural network increases exponentially with the number of dimensions  $n$  of the input vectors  $x_i$ . A larger number of weights requires more training examples before the network learns the correct function. As a result, neural networks used with even modest-sized input vectors often require many more training examples than are available.

Moreover, we do not have any performance guarantees for neural networks the way we have for the perceptron. However, the main advantage of the neural network over the perceptron is that it can learn non-linear functions (see Figure 1.5).



**Figure 1.5. Non-Linearly Separable Data Set with Curve Separating  
the Two Types of Data.**

In order to minimize the number of examples to train a neural network, we remap the dataset to feed into the network hand-crafted features. For example we can remap the “apples and oranges” dataset so that color information is the most important feature, since most oranges are orange and most apples are yellow, green or red. If we use the color information as the new input to the neural network, then the network trains faster, i.e. with fewer examples, and with higher classification accuracy. However, the process of finding the best set of features is not automated and it becomes increasingly difficult with larger sized problems.

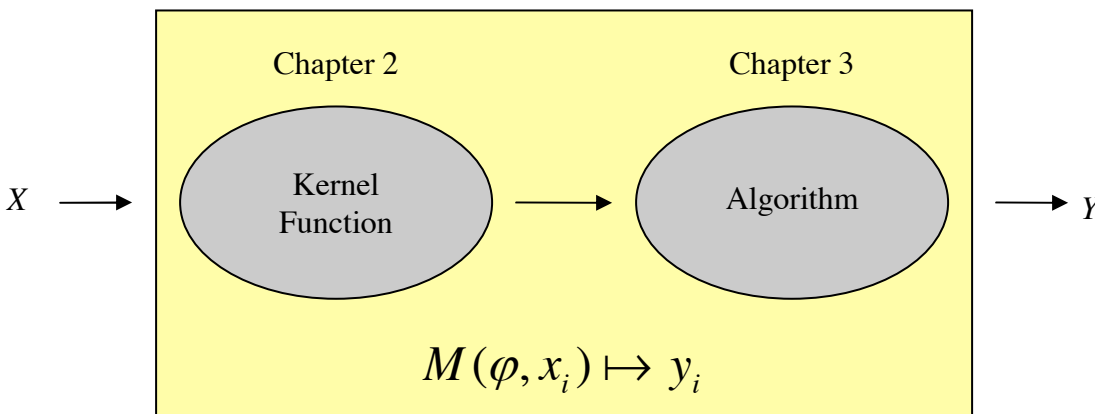
### 1.3.3 Kernel Methods

We use *kernel methods* to deal with the neural network’s deficiencies and also to keep its advantages. We discuss this topic more thoroughly in Chapters 2 and 3; however for now we briefly present what kernel methods allow us to do. Kernels have the best of both worlds: they are able to learn non-linear functions while they do so with a reasonable number of training examples. Using a *kernel function*, we project the data in a higher dimensional *feature space* that can potentially make the data linearly separable. Thus, the number of training examples is smaller since we use training algorithms similar

to those of perceptrons to learn the function  $M$  in this feature space. Moreover, one of the kernels' biggest advantages is that one can use several different kernel functions with the same algorithm, and several different algorithms with the same kernel function. We are able to quickly swap them around to run several experiments with little effort.

Thus, choosing a function  $M(\varphi, x_i) \mapsto y_i$  representing the supervised learning algorithm is a two step process: first we choose a kernel function, and then we choose an algorithm that uses the kernel function and learns the set of parameters  $\varphi$ . Different kernel functions specify different distance relationships, and implicitly features, between the input vectors, while different algorithms use the distance information from the kernel functions to learn the parameters  $\varphi$  that determine the final function  $M$ . We present kernel functions more thoroughly in Chapter 2 and the algorithms that use them in Chapter 3.

For now please refer to Figure 1.6 to understand how  $M$  is constructed.



**Figure 1.6. Overview of kernel approach to solving the supervised learning problem.**

In conclusion, kernel methods not only are easy to train and accurate but they are also flexible to use for large complicated learning tasks. Because of their advantages, the research community has heavily explored their use during the last decade.

The following table summarizes the three supervised learning classifiers:

	<b>Advantages</b>	<b>Disadvantages</b>
<b>Perceptrons</b>	<ul style="list-style-type: none"> <li>• Precise performance guarantees about future predictions</li> <li>• Small number of examples required to train</li> </ul>	<ul style="list-style-type: none"> <li>• Only work on linearly separable data</li> </ul>
<b>Neural Networks</b>	<ul style="list-style-type: none"> <li>• Work on non-linearly separable data</li> <li>• Work well in practice on difficult problems such as speech recognition and handwriting recognition</li> </ul>	<ul style="list-style-type: none"> <li>• Lack of exact performance guarantees about future classification performance</li> <li>• May require large number of training examples</li> <li>• Not suitable for larger problems</li> <li>• Hand-crafting features of input space to improve performance not automated and not always optimal</li> </ul>
<b>Kernels</b>	<ul style="list-style-type: none"> <li>• Precise performance guarantees</li> <li>• Small number of examples required to train</li> <li>• May work on non-linearly separable data (also see disadvantages)</li> <li>• Allow running several different linear algorithms in very high dimensional feature space</li> </ul>	<ul style="list-style-type: none"> <li>• Sometimes the high dimensional feature space cannot linearly separate data (also see advantages)</li> </ul>

**Table 1: Advantages and Disadvantages of Perceptrons, Neural Networks and Kernel Methods**

## 1.4 Problem Details and Technical Notations

In this paper we explore the possibility of computers to accomplish three tasks: Identifying a person from a set of photographs of people, identifying people with long hair versus people with short hair, and identifying people who are smiling versus people who are not smiling. For all tasks we use a database of photographs of human faces.

The database is composed of 590 grayscale images at a 92x112 resolution of people acting in different situations. Each photo is a column vector  $x_i$ , where  $1 \leq i \leq 590$ . Depending on the supervised learning task we try to accomplish, there is a rating  $y_i$  associated with each  $x_i$ , such as identifying a particular person, identifying people with long hair versus people with short hair, and showing whether a person is smiling. The set  $X = \{ x_i: 1 \leq i \leq 590 \}$  is the image database. Each algorithm learns a function  $M(\phi, x_i) \mapsto y_i$ . We compare the algorithms' classification percentage for each task.

## 1.5 Previous Work

Mathematicians and statisticians have used kernels since the early 1960s. The machine learning community embraced kernel methods only in the last decade. In 1962, Parzan was among the first mathematicians to use an early notion of kernels [13]. Aizerman in 1964 recognized the “kernel trick” for avoiding lengthy computations in high dimensional spaces. Boser in 1992 is generally credited with the creation of Support Vector Machines (SVMs) [2]. He was among the first machine learning researchers to use the idea of a kernel to form a feature space. At the time, neural network research was popular and people were struggling with the idea of crafting feature spaces for neural network learning problems. Several years later, in 1997, Scholkopf found that kernels could be used to increase support vector machine (SVM) applications in non-linear



domains [15, 17]. Soon after, the flood gates opened in terms of research with kernels applied to many known algorithms.

Historically, the three primary algorithms for face detection and feature identification have been Principal Components Analysis (PCA), Independent Component Analysis (ICA) and Linear Discriminant Analysis (LDA). These algorithms and techniques implement linear transformations (shearing, rotations and scaling). PCA is also known as eigen-value decomposition and Karhunen-Loeve transformation. PCA works by finding a new basis for a dataset such that the basis vectors are sorted by maximizing the variance in the data [4, 12]. ICA finds a new basis for the data such that the statistical dependencies in the data are minimized [1, 4]. LDA finds a basis such that the basis vectors are most capable of discriminating between the training categories [4, 10, 11]. Moreover, Support Vector Machines (SVMs) are used to find the hyper-plane that maximizes the distance separating the categories of points. Kernel methods can be combined with any of the above approaches in addition to any number of other algorithms to allow for non-linear transformations of the data. Due to the simplicity and performance of kernel methods, the research community has embraced them during the last four years. Thus, there have been hundreds if not thousands of papers published on face recognition and feature detection, mostly during the last decade when computers became powerful enough and algorithms sophisticated enough [3].

## **1.6 Paper Outline**

Throughout the rest of the paper we discuss in detail how we can solve certain problems regarding human faces using several kernel methods. In the Kernels chapter we discuss what kernels are, their properties, and how they help us solve some interesting

problems. The Algorithms chapter deals with the algorithms we employ with our kernels and what they allow us to achieve. Chapter 4, Our Data, explains how we collected and rated the images in our database and what format we use to represent them. Chapters 5, 6 and 7 present our experiments, their goal and their results, while in the last chapter we discuss the overall conclusion and mention some questions for future research.

# Chapter 2:

## Kernels

To review, we have classification problems for which we have to learn a function  $M$  that best maps the input to a desired output. In our case, the dimensionality of the input space is high, as we have  $92 \times 112 = 10304$  pixels for each image, so each image  $x_i \in \mathfrak{R}^n$ , where  $n = 10304$ . For the tasks typical of face and feature recognition, the points or images are not linearly separable in input space. As stated in the previous chapter, we need the input vectors to be linearly separable so we can use linear algorithms to learn the categories. Kernels are an automatic method to linearly separate the input data so that we can train linear algorithms on it.

### **2.1 Remapping Input Space to Feature Space**

For almost all tasks we need a method to remap the input space into a *feature space*  $F$  that will hopefully make the data linearly separable (see Figure 2.1). In other words, we need a mapping  $\phi$  defined as

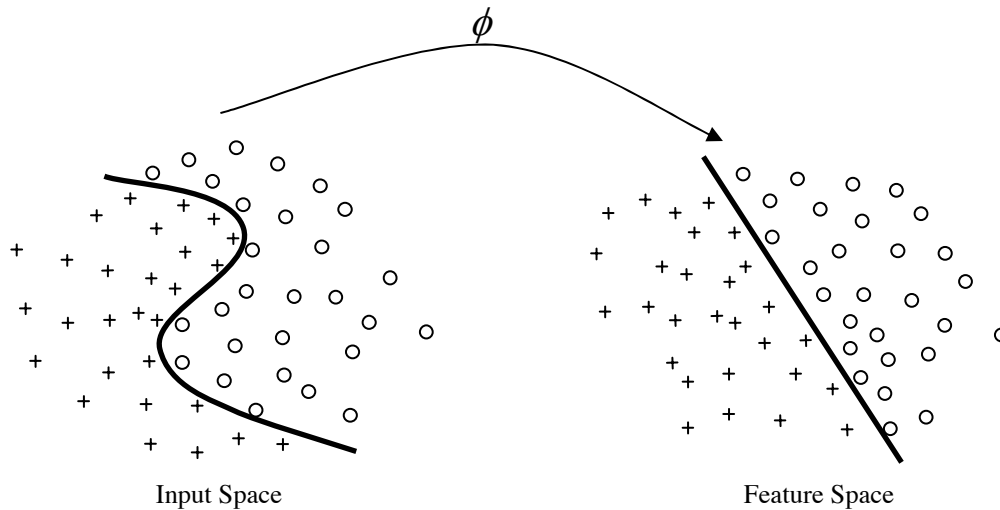
$$\phi: x \mapsto \phi(x) \in F. \quad (2.1)$$

For Neural Networks, researchers typically handcraft the mapping  $\phi$  using expert domain knowledge. This step is usually necessary to reduce the dimensionality of the

input space so that the Neural Network does not require a large number of training samples.

Feature spaces with kernel methods are different. First the feature space is not handcrafted. Instead it is determined by the selection of a kernel. Different kernels implicitly indicate different feature spaces. Second, the feature space is usually of a much higher dimensionality so that the data is more easily linearly separable.

The performance of kernel methods depends on how easy it is to linearly separate the data and also the algorithm employed to learn the separation. The choice of the right kernel for a particular data distribution is crucial in making the data linearly separable. In section 2.3 we see how a particular kernel can linearly separate in feature space a particular kind of data distribution in input space.



**Figure 2.1.** The function  $\phi$  remaps the input data into

**a feature space where it is linearly separable.**

## 2.2 Definition of a Kernel

*Definition of a kernel:* A kernel is a function  $k$  that for all  $x, z \in \mathfrak{R}^n$  satisfies

$$k(x, z) = \langle \phi(x), \phi(z) \rangle, \quad (2.2)$$

where  $\phi$  is defined in (2.1) [14].

Intuitively a kernel is a function that computes how similar two vectors are. The closer the vectors are to each other the higher the value of the kernel for those two vectors. Formally a kernel is a mapping from  $\mathfrak{R}^n \times \mathfrak{R}^n$  to  $\mathfrak{R}$  given by  $(x, z) \mapsto k(x, z)$ , where  $x, z \in \mathfrak{R}^n$  and  $k(x, z) \in \mathfrak{R}$ .

A special example of a kernel function of two vectors is the dot product:

$$k(x, z) = \langle x, z \rangle. \quad (2.3)$$

Note that  $\phi(x) = x$  in this case. Thus this kernel function computes distances in input space rather than feature space.

## 2.3 The Direct Method

The direct method is one approach to computing the kernel function [14, 16]. As an example consider a two-dimensional input space  $X \subseteq \mathfrak{R}^2$  and a feature mapping

$$\phi : x = (x_1, x_2) \mapsto \phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in F = \mathfrak{R}^3. \quad (2.4)$$

We compute the kernel function for this feature mapping in the following way:

$$\begin{aligned} k(x, z) &= \langle \phi(x), \phi(z) \rangle \\ &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \rangle \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \end{aligned} \quad (2.5)$$

Thus, we compute  $\phi(x)$ , then  $\phi(z)$ , and then compute their dot product. The problem with this method is the number of calculations. If  $\phi$  is a mapping from a lower

dimensional input space to a very high dimensional feature space, the computations of  $\phi(x)$  and  $\phi(z)$ , and their dot product may require a very long time to finish.

The function  $\phi$  in 2.4 maps the data in Figure 2.2 from its input space on the left to the feature space on the right. Although the feature space is technically three-dimensional, we represent it in two dimensions for simplicity of visualization. Note how the data becomes linearly separable in feature space. The equation of the ellipse (in input space) that forms the decision boundary is linear in feature space. This is the critical role that kernels play in classification tasks.

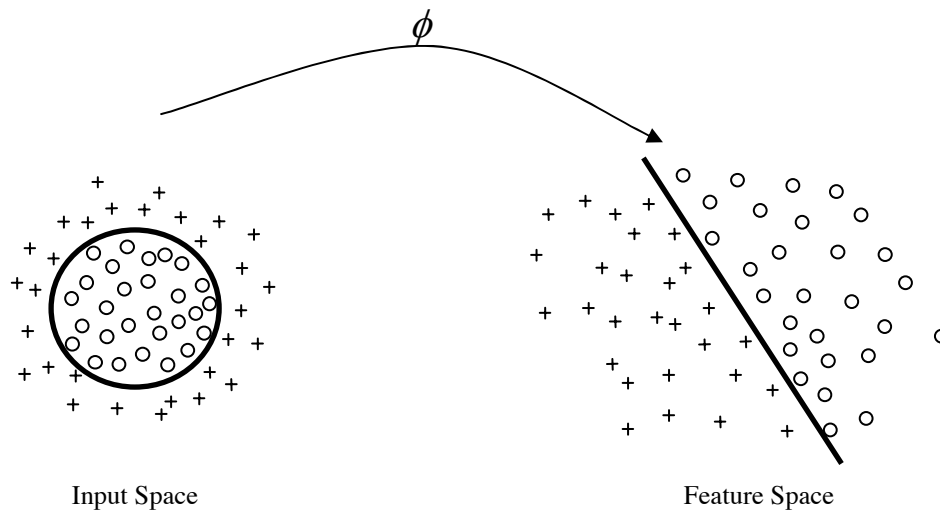


Figure 2.2. Linearly separating the data using the mapping  $\phi$  defined in 2.4.

## 2.4 The Kernel Trick

There is a different approach to computing the kernel function that speeds up computation time considerably. Consider the same two-dimensional input space  $X \subseteq \mathbb{R}^2$  and feature mapping (2.4). Then according to (2.5) the kernel function is given by

$$k(x, z) = x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2 .$$

Note that

$$x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2 = (x_1 z_1 + x_2 z_2)^2 = \langle x, z \rangle^2,$$

and hence

$$k(x, z) = \langle x, z \rangle^2$$

is a kernel function for the feature mapping  $\phi$ . The last computation is clearly much faster than computing the coordinates of each vector in feature space and then taking the dot product. The kernel trick allows us to avoid computing the coordinates in feature space of each input vector explicitly and it gives the result directly into the feature space [14, 16].

Also note that the feature space is not uniquely determined by the kernel function.

For

$$\phi : x = (x_1, x_2) \mapsto \phi(x) = (x_1^2, x_2^2, x_1 x_2, x_1 x_2) \in F = \mathfrak{R}^4$$

we have the same kernel function  $k(x, z) = \langle x, z \rangle^2$ . For simplicity we usually choose a particular kernel function and consider one of the feature mappings for that kernel function as our feature space.

## 2.5 The Kernel Matrix

To simplify data representation and also to better visualize kernels, we make use of the *kernel matrix* notation. Given a finite dataset  $S = \{x_1, \dots, x_l\} \subset \mathfrak{R}^n$ , we represent all possible values of the kernel function for the dataset in the following kernel matrix notation:

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & k(x_1, x_3) & \dots & k(x_1, x_l) \\ k(x_2, x_1) & k(x_2, x_2) & k(x_2, x_3) & \dots & k(x_2, x_l) \\ k(x_3, x_1) & k(x_3, x_2) & k(x_3, x_3) & \dots & k(x_3, x_l) \\ \dots & \dots & \dots & \dots & \dots \\ k(x_l, x_1) & k(x_l, x_2) & k(x_l, x_3) & \dots & k(x_l, x_l) \end{bmatrix} \text{ and}$$

$$K_{ij} = k(x_i, x_j), \text{ where } i, j = 1..l.$$

Moreover, the algorithms from Chapter 3 use the kernel matrix to improve performance by avoiding the re-computation of the kernel function on the input data.

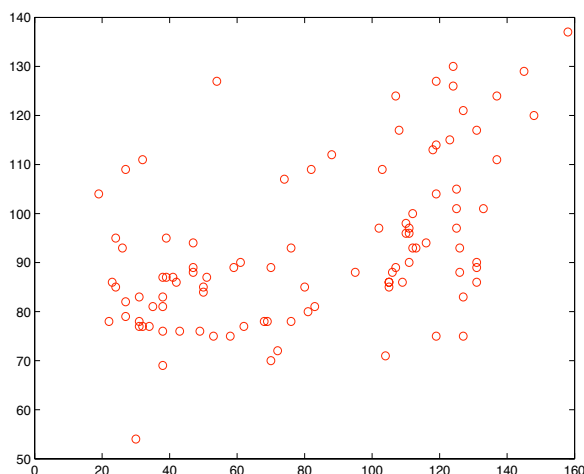
## 2.6 Examples of Kernels

### 2.6.1 The Inner Product

We have already seen this kernel in section 2.2:

$$k(x_i, x_j) = \langle x_i, x_j \rangle.$$

This is the simplest kernel function and it can be used with algorithms to learn on linearly separable data where the feature space is the input space. The reason is that the feature mapping function  $\phi(x) = x$  is just the identity function.

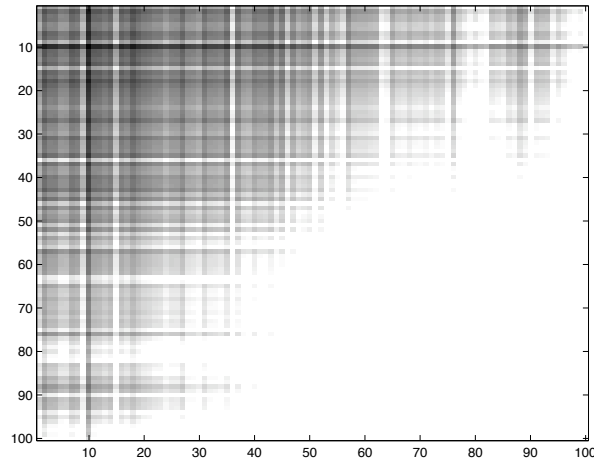


**Figure 2.3. 100 2-dimensional dataset.**

In Figure 2.3 we represent a dataset composed of 100 elements and in Figure 2.4 the



kernel matrix associated with the above kernel function and dataset represents a picture for which each pixel is the corresponding value of the kernel matrix. In the kernel matrix the points are sorted according to their  $x$ -axis.



**Figure 2.4. Inner product kernel of dataset.**

## 2.6.2 Polynomials of Inner Products

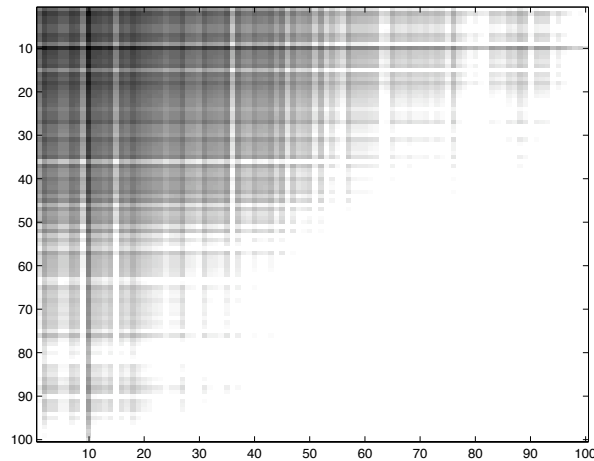
The general form of these kernels is:

$$k(x_i, x_j) = (\langle x_i, x_j \rangle + c)^m,$$

where  $m$  and  $c$  are parameters. When  $m = 1$  and  $c = 0$  we have the degenerate case of the inner product. We have already seen the case when  $m = 2$  and  $c = 0$ :

$$k(x_i, x_j) = \langle x_i, x_j \rangle^2. \tag{2.6}$$

Polynomial kernels allow working in a higher dimensional space, such as the feature space given by (2.4). As we have seen in Figure 2.2, by mapping the input data into these higher dimensional feature spaces we are sometimes able to make the data linearly separable. In Figure 2.5 we depict the kernel matrix for the kernel function (2.6) for the dataset in Figure 2.3.



**Figure 2.5. Squared inner product kernel of dataset.**

**Notice the similarity to the input space kernel.**

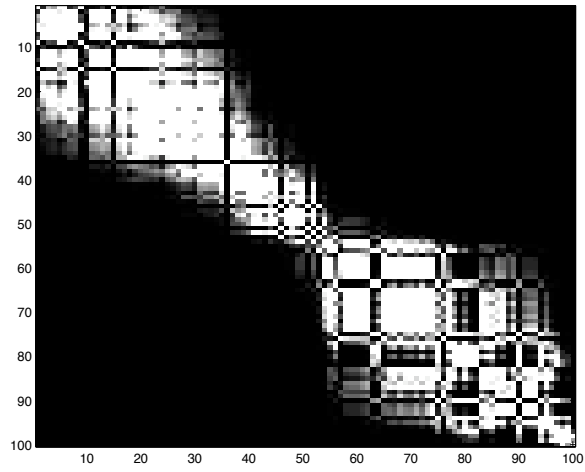
### 2.6.3 The Gaussian Kernel

This kernel function is:

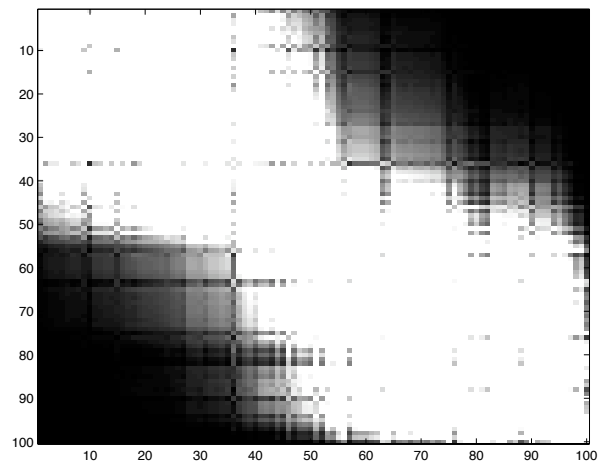
$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right),$$

where  $\sigma > 0$  is called the width parameter. Note that  $k(x, x) = \exp(0) = 1$ . The Gaussian kernel specifies an infinite-dimension feature space where higher order dimensions decay faster than lower order dimensions.

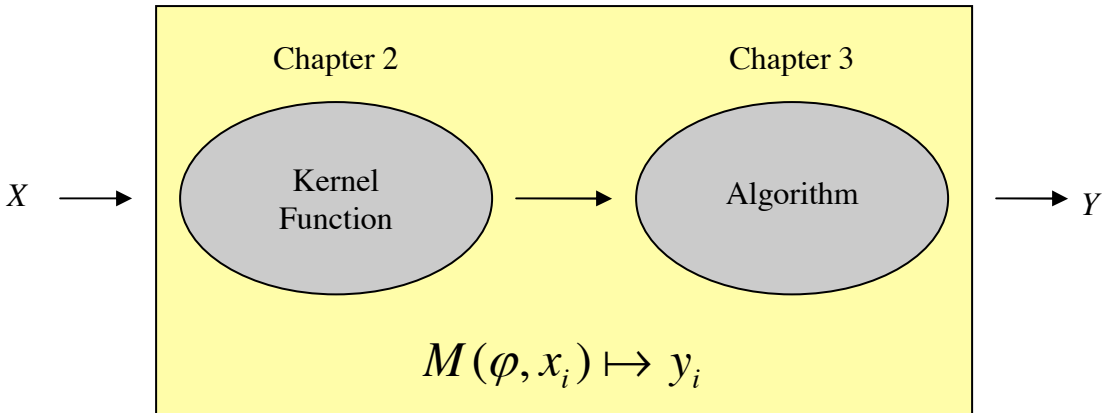
In Figure 2.6 we show the picture representation of the Gaussian kernel matrix with a width  $\sigma = 10$  for the dataset in Figure 2.3, and in Figure 2.7 the same kernel but with a width of  $\sigma = 30$ .



*Figure 2.6. Gaussian kernel with  $\sigma = 10$ .*



*Figure 2.7. Gaussian kernel with  $\sigma = 30$ .*



**Figure 2.8. Overview of kernel approach to solving the supervised learning problem.**

So far we have seen how kernels implicitly define a feature space, how they can be computed and how different kernel functions determine different feature spaces. In Figure 2.8 we show again the overall kernel method approach to solving the supervised learning problem. In the next chapter we examine how algorithms use kernels and the implicit feature mappings to learn the function  $M(\varphi, x_i) \mapsto y_i$ .

# Chapter 3: Algorithms

In this chapter we focus on the algorithms we use in our experiments. We start with a review of how kernels are used in supervised learning algorithms, and then talk about the way the data is divided in training and testing sets. We give a brief description of each of the supervised learning algorithm and finally discuss an algorithm that helps us to both visualize and remove noise from data. We give only brief, intuitive discussions of each algorithm. For more information, see the respective chapters in [14].

## **3.1 Role of Kernels**

The supervised learning algorithms we use require distances between points for determining their “similarity” to one another. The distance between points can be measured in the original input space, or in a different feature space. The advantage of computing distances of points in feature space is that the data may be linearly separable.

We use the kernel functions to compute distances in feature space between all points without explicitly computing the mapping to feature space  $\phi$ . Given two vectors in feature space  $\phi(x)$  and  $\phi(z)$ , we compute the distance between each of them by finding the norm of the vector  $\phi(x) - \phi(z)$ .

$$\begin{aligned}
\|\phi(x) - \phi(z)\|^2 &= \langle \phi(x) - \phi(z), \phi(x) - \phi(z) \rangle \\
&= \langle \phi(x), \phi(x) \rangle - 2\langle \phi(x), \phi(z) \rangle + \langle \phi(z), \phi(z) \rangle \\
&= k(x, x) - 2k(x, z) + k(z, z),
\end{aligned}$$

where on the last line we have used the substitution (2.2). This procedure relies on the kernel trick in section 2.4. We are able to compute distances between vectors in feature space, just by using the kernel function, thus avoiding the explicit, potentially time consuming transformation  $\phi$ .

By using kernel functions, the algorithms learn a function  $M(x) \mapsto y$ , where  $x$  is the input vector and  $y$  is the desired categorization of  $x$ . The algorithms store two kernel matrices to reduce running time by avoiding re-computation of the kernel function. One matrix corresponds to points used to *learn* the function  $M$ , and another matrix is corresponding to new points used to *test* the learned function  $M$  against the correct classification. Thus, the data is divided into testing and training sets.

### 3.2 Datasets

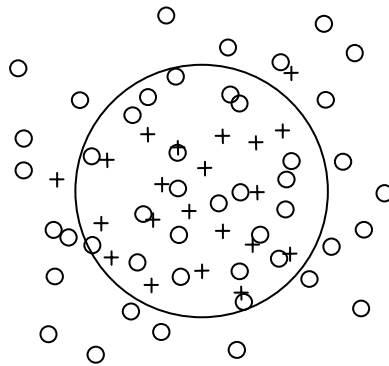
The data for all supervised learning algorithms is divided into training sets and testing sets. The training sets are composed of pairs of the form  $(x, y)$ , where  $x$  represents a training vector and  $y$  is the correct classification of  $x$ . The algorithms use the correct category of each  $x$  to learn the function  $M$ , so that  $M(x) = y$ . The testing sets are of the same form as the training sets, however  $y$  is used to compute the performance of the algorithm by comparing the output of the algorithm on each testing vector  $x$  to the correct classification  $y$ . The data should be distributed similarly in both the testing and training sets so that the classification algorithm does not learn biases in the training set.

### 3.3 Distance Algorithms

Distance algorithms use the distance information between points directly to learn the function  $M$  and also to determine the classification of new points. We review three algorithms here.

#### 3.3.1 Naïve Novelty Detector

This algorithm requires the training set be composed of points from only the category that needs to be distinguished from the other categories. The algorithm statistically finds the center of mass and a radius of an encompassing hyper-sphere of the training set such that the training points are included. The algorithm has an adjustable parameter  $\delta$ , called a confidence parameter, to allow for extreme training points to be excluded from the hyper-sphere. After the training stage, the hyper-sphere is used as a boundary to decide if a testing point is novel, i.e. outside the hyper-sphere, or not (see Figure 3.1).



**Figure 3.1. Naïve Novelty Detection: + denotes training points, o denotes testing points.**

**Note how some training points have been excluded from the hyper-sphere.**

Note that in order to decide the position of the center of mass as well as the radius, the algorithm needs to know the distances between the training points. Moreover, the distances between the testing points and the center of mass determine whether the testing points lie outside the hyper-sphere, i.e. whether the distance to the center of mass

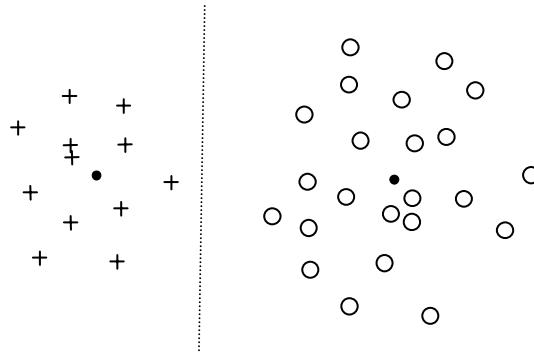
of each testing point is bigger than the radius of the hyper-sphere. This is the purpose of the distance information given by the kernel. By using a kernel instead of computing the distances explicitly, the algorithm provides the same functionality in various feature spaces as well as in input space.

### **3.3.2 Parzan Window Classifier**

The training set for this algorithm is composed of two or more categories of points. The goal of the algorithm is to learn to classify testing points into categories. The algorithm computes the center of mass for each of the training categories and adjusts the decision boundaries according to the number of points in each of these categories. Thus, if a category has more points than others, the decision boundary is moved farther away from the center of mass of that category in a manner similar to the prior probabilities in Bayesian calculations. Every testing point has its distance to each of the center of mass computed and then is categorized depending on which side of the decision boundaries it is.

An example is given in Figure 3.2 using two categories in a 2D space. Notice how the decision boundary is shifted slightly to the left of the half-way point because there are more o's than +'s. Every new point lying to the left of the decision boundary is categorized as a +, while every point on the right of the boundary is categorized as a o.





**Figure 3.2. Parzan Window Classifier: Note the shift to the left of the decision boundary.**

**The solid dots indicate the centers of mass of each category.**

### 3.3.3 Fisher Linear Discriminant Classifier

This algorithm is similar to the Parzan Window Classifier, except that it also takes into account the spread of the data in each category. Thus, if a category has a very high density, the decision boundary is closer to the center of mass of the category. On the other hand, if a category has many spread out points, the decision boundary is farther away from the center of mass of the category to account for the high variance of the data, and thus encompass future, probably highly variable testing points.

In Figure 3.3 we have two categories of data represented in a 2D space. Note that the '+'s are now many more but also more highly concentrated than the 'o's. The figure shows the difference between the Parzan Window Classifier and the Fisher Linear Discriminant Classifier decision boundaries in this situation.

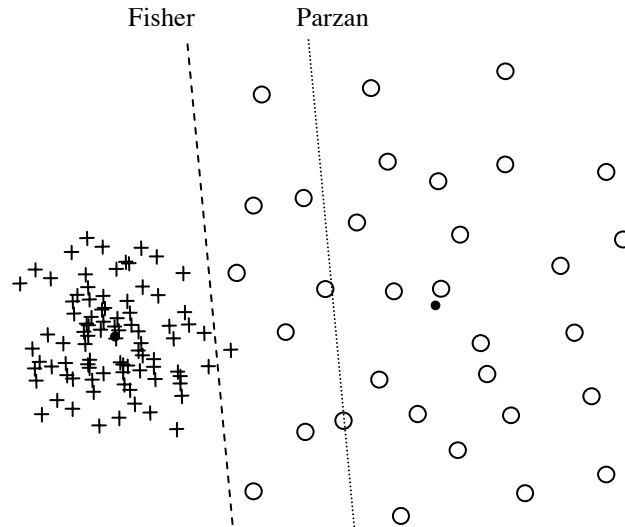


Figure 3.3. Fisher Linear Discriminant Classifier (FLDC) vs. Parzan Window Classifier (PWC).

### 3.4 Support Vector Machines

Support vector machines (SVMs) are a class of algorithms that use only key vectors from the training set to determine the decision boundary. These vectors are called support vectors. The idea behind using only a subset of the training set to contribute to the decision boundary is to limit the number of computations between the training vectors and the testing vectors. If we determine the key training vectors for the decision boundary, then we completely eliminate all the other vectors in the training set from the computations involved when testing new points.

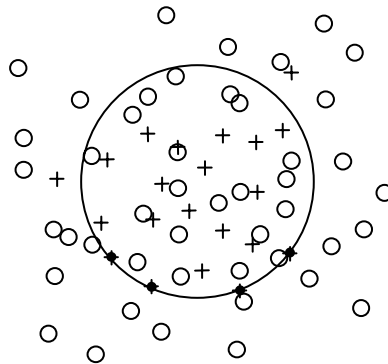
Moreover, one can exclude some extreme training vectors from the decision making task, so that the decision boundary is not affected by extraneous, noisy data. The amount of “sloppiness” is determined by a parameter  $\nu$  and thus the technique is called the  $\nu$ -trick.

We review two algorithms that make use of support vectors.

### 3.4.1 SVM Novelty Detector

This algorithm is similar to the Naïve Novelty Detector. It uses only support vectors for its decision boundary, thus the boundary is tighter around the training points and more easily fits the input data.

In Figure 3.4, note how the algorithm allows for excluding many training points from the decision task, since only the support vectors are used for the testing of new points. This procedure speeds up testing considerably, especially if the training set is large.

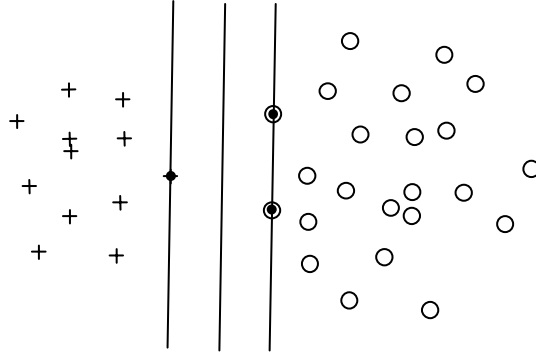


**Figure 3.4. SVM Novelty Detection: + denotes training points, o denotes testing points, a solid dot denotes support vectors.**

### 3.4.2 SVM Maximum Margin Classifier

This algorithm finds the hyper-plane that best separates two classes of points. The support vectors are the vectors from each category that directly influence the decision boundary.

We provide an example in Figure 3.5. Note how only the support vectors are used when testing for new points. They are the only vectors contributing to the decision boundary.



**Figure 3.5. SVM Maximum Margin Classifier.**

**The solid dots indicate the support vectors.**

### 3.5 Principal Components Analysis

Principal Components Analysis (PCA) is a technique by which data is linearly remapped to a new basis that best captures the variance in the data [8, 14]. Usually, the new basis vectors are ordered according to the variance of the data projected on that vector. PCA performs a linear mapping from input space to this new basis:

$$x_i \in \mathfrak{R}^n \mapsto \tilde{x} \in \mathfrak{R}^n$$

PCA is often used to reduce the dimensionality of the input space. We can use the first  $k$  dimensions of  $\tilde{x}$ ,  $1 \leq k \leq n$ , to capture most of the variance but have a smaller number of dimensions. This procedure sometimes helps learning algorithms perform better, since they can look only at the first  $k$  most “important” vectors.

Another use for PCA is for visualizing data. We make use of this technique to plot 2-dimensional vectors representing whole images. From  $n = 10304$  dimensions, we capture the highest variance of the data in the first  $k = 2$  dimensions and we plot the vectors after this transformation.

PCA is also used for denoising. Since many of the low-variance dimensions of  $\tilde{x}$  contain very little variance in the data, by intentionally excluding some of the

low-variance dimensions, we remove extraneous noise from the input data, thus improving performance.



# Chapter 4: Our Data

## **4.1 Human Face Database**

For our experiments we custom-built a database of images of various people's faces in several moods. Each photograph is an 8-bit grayscale image of 92 by 112 pixels of the cropped face of a person. We have 590 such photographs. The first 200 pictures are of 5 people, 40 pictures each, while the remaining 390 pictures are of 39 people, 10 pictures each. The people show different moods and face expressions that were acted during the photo shoot as several situations were portrayed. The photos have a neutral white background and are taken in a controlled light environment. They are represented as column vectors in Matlab and they are all placed together in a 10304x590 matrix. When used with the learning algorithms, the photos are divided into training and testing sets depending on the task and the algorithm used.

The photos are arranged in sets of 10. Five people have four sets ( $4 \times 10 = 40$  pictures per person), while 39 people have one set each. Each photo in a series corresponds to a particular situation. The following is the list of situations we presented to the people:

1. Act as if you are in a happy mood (you are content with your life).

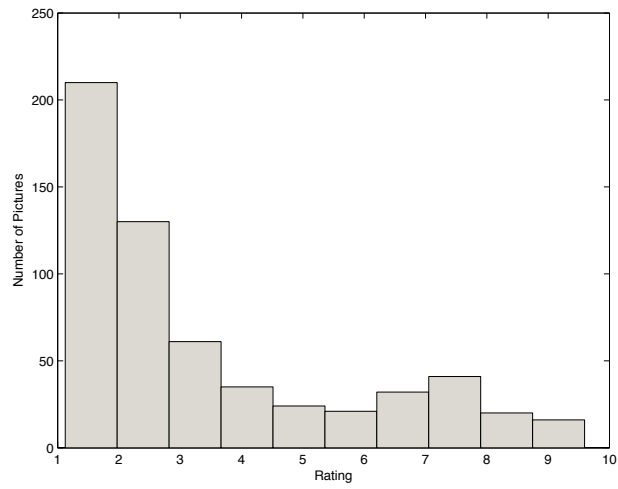
2. Act as if you just heard something funny.
3. Act as if you are excited to leave tomorrow on the best vacation ever.
4. Act as if you received a bad grade today.
5. Act as if your favorite pet died.
6. Act as if you are mad at your supervisor for not treating your work fairly.
7. Act as if you are scared of darkness and you heard something weird somewhere in the corridor.
8. Act as if someone just startled you to death and now you are catching your breath.
9. Act as if you are indifferent and do not care much about the things around you.
10. Make a funny face.

We present the whole database in Appendix A. The numbers of the above situations correspond to the pictures within a set. Thus, the first picture in a set corresponds to the first question, the second picture to the second question and so forth.

## **4.2 Ratings**

We have a set of ratings from 1 to 10 showing how much each person is smiling in each photo, 1 being not smiling at all and 10 being smiling very much or laughing. The ratings are averages from several people who rated all or part of the image database. The histogram of the ratings is presented in Figure 4.1. We use both a cut-off point of 6 and 8 in our experiments to see the differences in performance between them for the smiling detection task.





**Figure 4.1. Histogram of smile ratings.**



# Chapter 5:

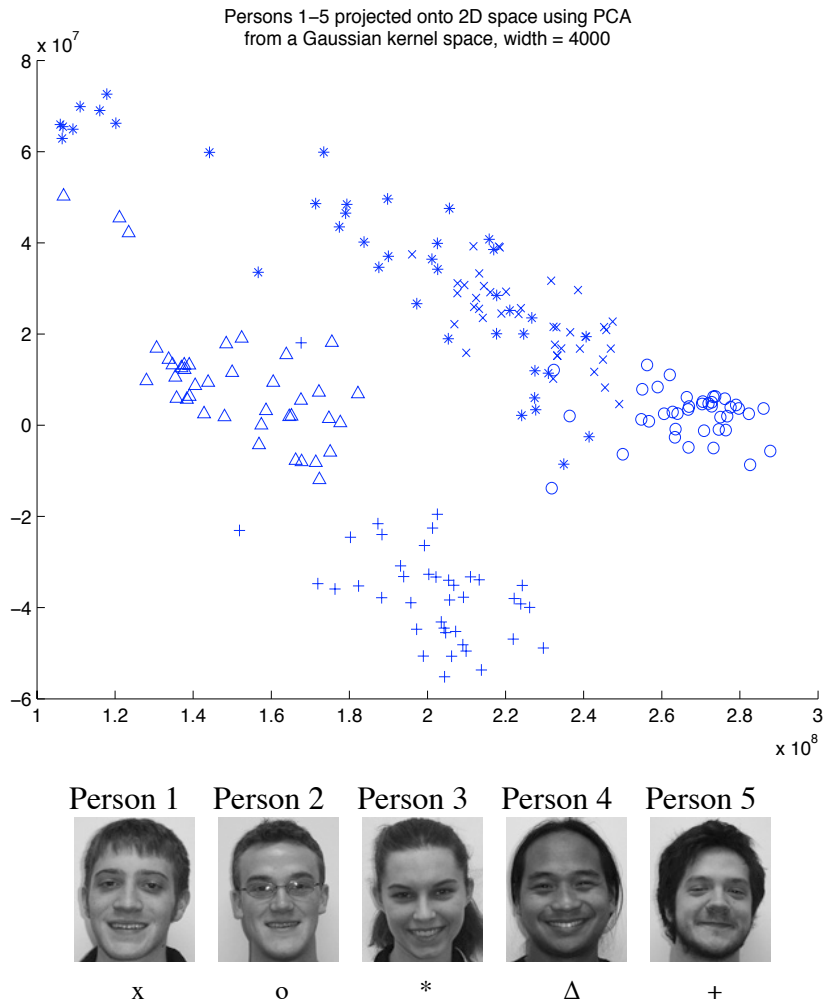
## Face Recognition

Our first set of experiments in the paper constitutes a face recognition task. The goal of this task is to identify one particular person from a set of photos of various people. We make references to several people from the database. Some pictures are included here for convenience. For the full database of photos, please refer to Appendix A.

Throughout our experiments we use the feature space defined by the Gaussian kernel. In order to better visualize what happens to the data after remapping into feature space we apply principal components analysis (PCA). Recall from section 3.5 that PCA is a technique that remaps the data to a new basis that captures most of the variance in each dimension. We use PCA in feature space as a visualization tool to map the data down to a two-dimensional space. We choose the two dimensions that capture most of the variance of the data so that we have a rough idea of the distances between the data points.

For the face recognition task the dataset is composed of the first 200 images of the database. To visualize the “position” of these images in feature space, we take the dataset, map each image into the feature space determined by a Gaussian kernel with a

width of 4000 and finally project it down onto the two dimensions that capture most of the variance given by PCA. Figure 5.1 gives a rough idea of the distribution of this dataset.



**Figure 5.1. Distribution of Face Recognition dataset in Gaussian kernel space after PCA.**

We conduct experiments for identifying Person 2 and Person 3. There are ten experiments in total, five for identifying Person 2 and five for identifying Person 3. Within a set of five experiments, each corresponds to one of the five algorithms we use: Naïve Novelty Detection, Parzan Window Classifier, Fisher Linear Discriminant Classifier, SVM Novelty Detection, and SVM Maximum Margin Classifier. We use a

Gaussian kernel with widths of 3000, 4000 and 6000 throughout and a width of 2000 for particular experiments.

For each experiment we provide the goal, the dataset used, the training set, the testing set, the type of kernel, the algorithm used, and the set of parameters for the algorithm. We present results for each set of parameters in a table specifying the kernel width used, the classification percentage, the number of testing points, the number of correct classifications, and a break down of false positives and false negatives. We conclude each experiment with a discussion of the results.

We discuss major conclusions of the face recognition task at the end of the chapter.

**Experiment 1.** Naïve Novelty Detection for Person 2.

**Purpose:** Identify pictures of Person 2 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 20 images of Person 2, first 5 from each series of 10 (Negatives).

**Testing Set:** the 20 other images of Person 2 (Negatives) and 160 images of 4 other people (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** Naïve Novelty Detection.

**Parameters:**  $\delta = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.1111	180	20	0	160
4000	0.1111	180	20	0	160
6000	0.1111	180	20	0	160

**Notes:** The algorithm classifies all pictures as being Person 2. We assume that the very high dimensionality of the input space is affecting the algorithm in the training part, so it forms a large radius that contains all training and testing points.

**Experiment 2.** Naïve Novelty Detection for Person 3.

**Purpose:** Identify pictures of Person 3 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 20 images of Person 3, first 5 from each series of 10 (Negatives).

**Testing Set:** the 20 other images of Person 3 (Negatives), and 160 images of 4 other people (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** Naïve Novelty Detection.

**Parameters:**  $\delta = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.1111	180	20	0	160
4000	0.1111	180	20	0	160
6000	0.1111	180	20	0	160

**Notes:** (Similar conclusion to Experiment 1). It seems the algorithm classifies all pictures as being Person 3. We assume that the very high dimensionality of the input space is affecting the algorithm in the training part, so it develops a very large radius that contains all training and testing points.

**Experiment 3.** Parzan Window Classifier for Person 2.

**Purpose:** Identify pictures of Person 2 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 100 images: first 5 images from each series (20 series in all).

Person 2 classified as +1 (Positive), rest of images classified as -1 (Negative).

**Testing Set:** 100 images: last 5 images from each series (20 series in all).

**Kernel:** Gaussian kernel.

**Algorithm:** Parzan Window Classifier.

**Parameters:** none.

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.20	100	20	80	0
4000	0.20	100	20	80	0
6000	0.20	100	20	80	0

**Notes:** The algorithm classifies all pictures as being Person 2. Here we have a similar phenomenon as in Experiment 1 and 2.



**Experiment 4.** Parzan Window Classifier for Person 3.

**Purpose:** Identify pictures of Person 3 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 100 images: first 5 images from each series (20 series in all).

Person 3 classified as +1 (Positive), rest of images classified as -1 (Negative).

**Testing Set:** 100 images: last 5 images from each series (20 series in all).

**Kernel:** Gaussian kernel.

**Algorithm:** Parzan Window Classifier.

**Parameters:** none.

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.20	100	20	80	0
4000	0.20	100	20	80	0
6000	0.20	100	20	80	0

**Notes:** (Similar conclusion to Experiment 3). The algorithm classifies all pictures as being Person 3. Here we have a similar phenomenon as in Experiment 1 and 2.

**Experiment 5.** Fisher Linear Discriminant Classifier for Person 2.

**Purpose:** Identify pictures of Person 2 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 100 images: first 5 images from each series (20 series in all).

Person 2 classified as +1 (Positive), rest of images classified as -1 (Negative).

**Testing Set:** 100 images: last 5 images from each series (20 series in all).

**Kernel:** Gaussian kernel.

**Algorithm:** Fisher Linear Discriminant Classifier.



**Parameters:**  $\lambda = 0.1$ .

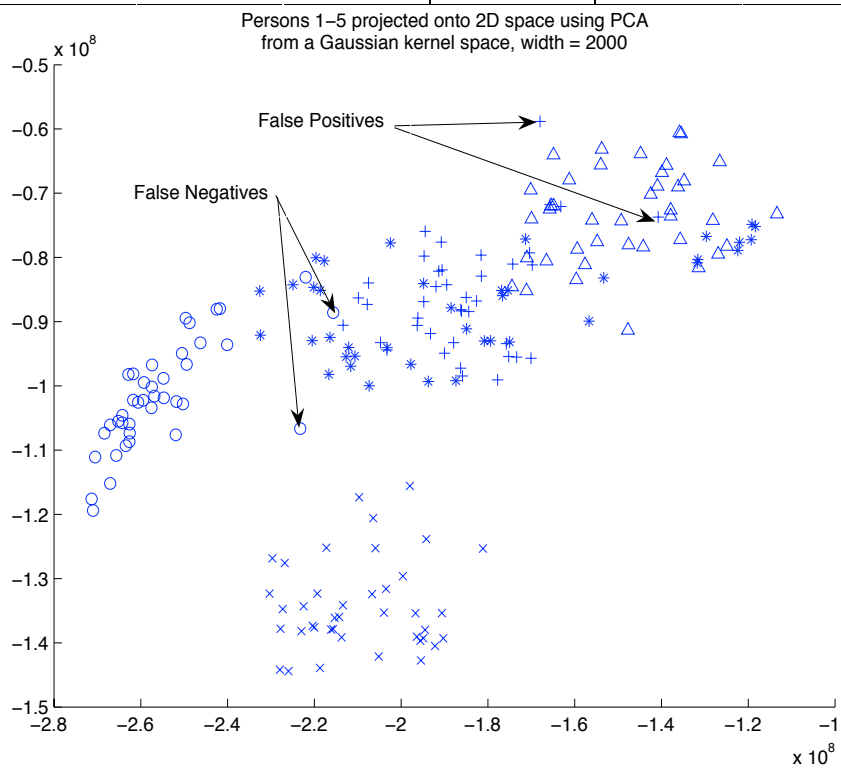
<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
2000	0.96	100	96	2	2
3000	0.95	100	95	5	0
4000	0.94	100	94	6	0
6000	0.92	100	92	7	1

**Parameters:**  $\lambda = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
2000	0.96	100	96	2	2
3000	0.95	100	95	5	0
4000	0.92	100	92	7	1
6000	0.91	100	91	8	1

**Notes:** There does not seem to be any significant difference between  $\lambda = 0.01$  and  $\lambda = 0.1$ , but the latter does give slightly better results. Otherwise the algorithm is performing quite well. A Gaussian kernel width of 2000 is the best. The two false negatives are probably because Person 2's face is turned very sharply from the regular pose. The two false positives are likely due to a similar reason. Person 5 is facing in some different direction and his vector representation may be closer to Person 2 in these two instances (see Figure 5.2).

<p>False Positives:</p>	 <p>167 +      190 +</p>	<p>False Negatives:</p>	 <p>48 o      77 o</p>
-------------------------	---	-------------------------	---



**Figure 5.2. Experiment 5 misclassifications.**

**Experiment 6.** Fisher Linear Discriminant Classifier for Person 3.

**Purpose:** Identify pictures of Person 3 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 100 images: first 5 images from each series (20 series in all).

Person 3 classified as +1 (Positive), rest of images classified as -1 (Negative).

**Testing Set:** 100 images: last 5 images from each series (20 series in all).

**Kernel:** Gaussian kernel.

**Algorithm:** Fisher Linear Discriminant Classifier.

**Parameters:**  $\lambda = 0.1$ .

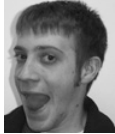










<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
2000	0.88	100	88	12	0
3000	0.89	100	89	11	0
4000	0.87	100	87	12	1
6000	0.86	100	86	13	1

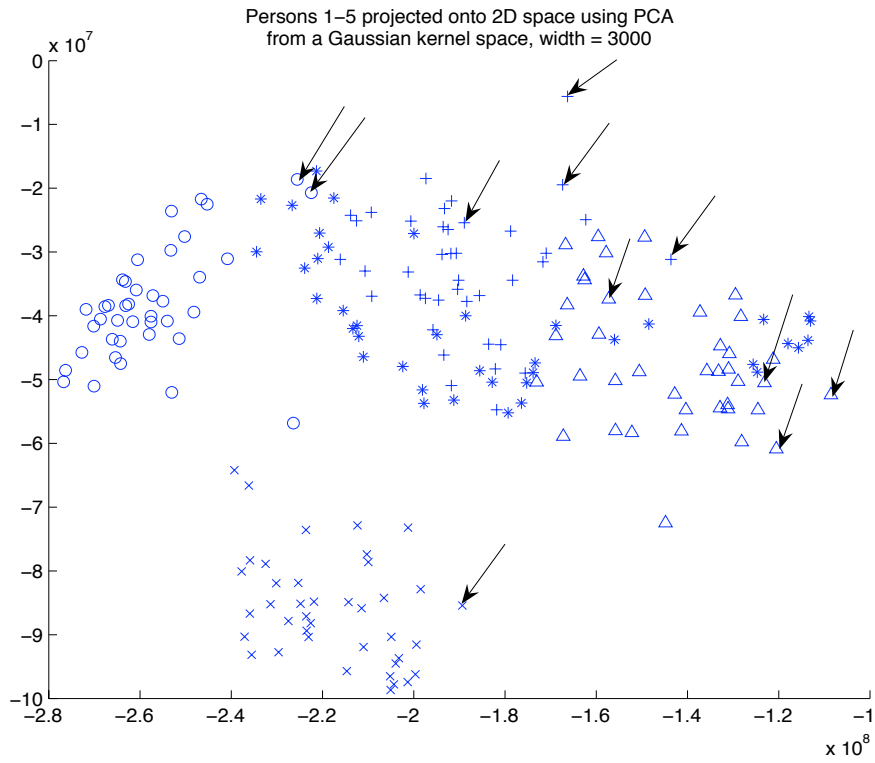
**Parameters:**  $\lambda = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
2000	0.88	100	88	12	0
3000	0.89	100	89	11	0
4000	0.87	100	87	12	1
6000	0.86	100	86	12	2

**Notes:** There is not a significant difference between  $\lambda = 0.01$  and  $\lambda = 0.1$ .

Otherwise the algorithm has average performance, considering that Person 3's photos are more varied than Person 2's since she has her hair both pulled up and untied (see Figure 5.3 and Appendix A). A Gaussian kernel width of 3000 is the best. We have no or very few false negatives but several false positives, which suggests the algorithm includes more pictures than just of Person 3 in the decision boundary. The false positives that we get are mostly photos of people looking in different directions or looking very different than their normal pose, such as making a funny face or looking downwards or to their left (see Figure 5.3).

False Positives:						
	10 x	48 o	60 o	130 $\Delta$	140 $\Delta$	150 $\Delta$
						
	160 $\Delta$	166 +	167 +	190 +	197 +	



**Figure 5.3. Experiment 6 misclassifications. Arrows denote false positives.**

**Experiment 7.** SVM Novelty Detection for Person 2.

**Purpose:** Identify pictures of Person 2 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 20 images of Person 2, first 5 from each series of 10 (Negatives).

**Testing Set:** the 20 other images of Person 2 (Negatives) and 160 images of 4 other people (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Novelty Detection.





**Parameters:**  $\gamma = 0.01, \nu = 0.1$ .

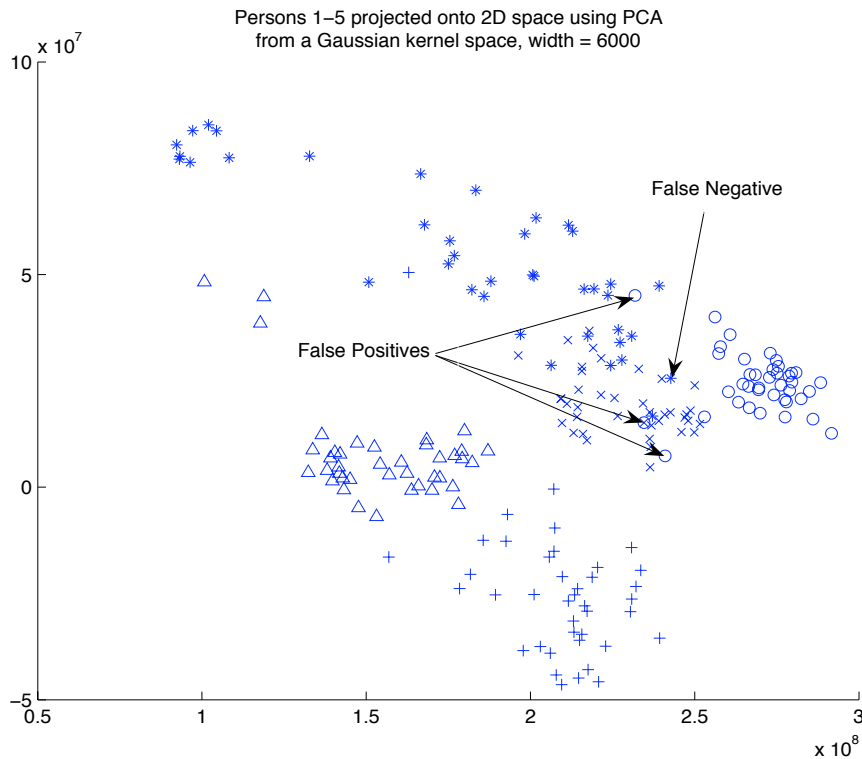
<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.91	180	165	15	0
4000	0.93	180	168	12	0
6000	0.95	180	171	9	0

**Parameters:**  $\gamma = 0.1, \nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.95	180	171	9	0
4000	0.97	180	175	5	0
6000	0.98	180	176	3	1

**Notes:** The algorithm performs consistently better when  $\gamma = 0.1$  as opposed to 0.01. For both settings a Gaussian kernel width of 6000 is best. For the best combination of gamma and kernel width, the three false positives are Person 2 again looking in a different direction (Image 48, 77) or being funny (Image 60). The only false negative is Person 3 being startled (Image 118).

False Positives:				False Negatives:	
	Image 48 o	Image 77 o	Image 60 o		Image 118 *



**Figure 5.4. Experiment 7 misclassifications.**



**Experiment 8.** SVM Novelty Detection for Person 3.

**Purpose:** Identify pictures of Person 3 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 20 images of Person 3, first 5 from each series of 10 (Negatives).

**Testing Set:** the 20 other images of Person 3 (Negatives) and 160 images of 4 other people (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Novelty Detection.




**Parameters:**  $\gamma = 0.01, \nu = 0.1$ .

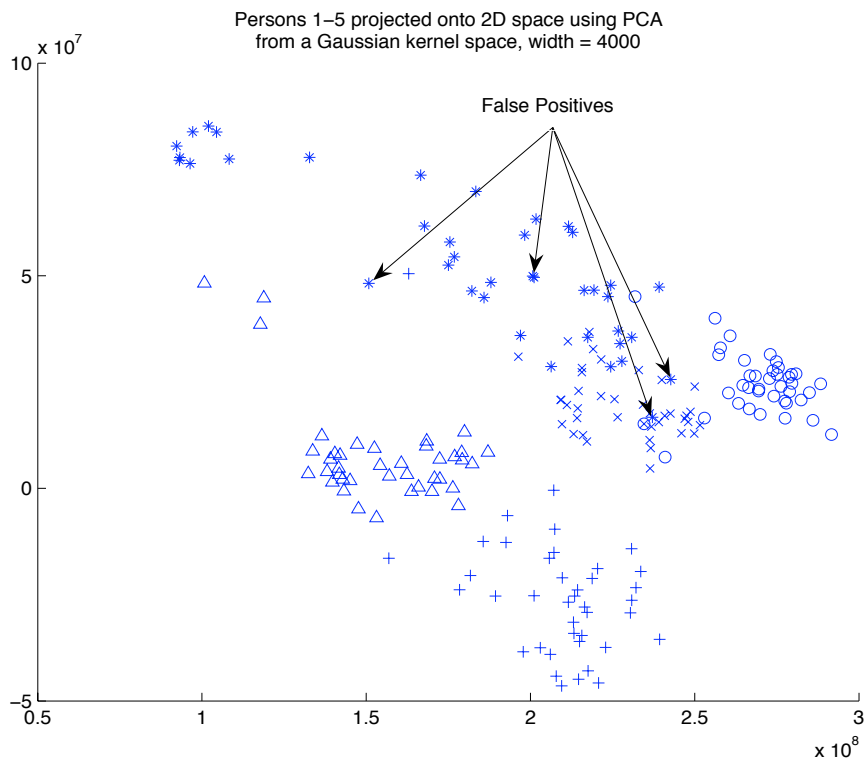
<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.91	180	163	17	0
4000	0.92	180	166	14	0
6000	0.93	180	168	12	0

**Parameters:**  $\gamma = 0.1, \nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.96	180	173	7	0
4000	0.98	180	176	4	0
6000	0.94	180	170	1	9

**Notes:** The algorithm performs consistently better with  $\gamma = 0.1$  as opposed to 0.01. A Gaussian kernel width of 4000 is best with  $\gamma = 0.1$ . For the best combination of gamma and kernel width, the four false positives we get are Person 3 looking different from her normal pose.

False Positives:				
	Image 88 *	Image 98 *	Image 110 *	Image 118 *



**Figure 5.5. Experiment 8 misclassifications.**

**Experiment 9.** SVM Maximum Margin Classifier for Person 2.

**Purpose:** Identify pictures of Person 2 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 100 images: first 5 images from each series (20 series in all).

Person 2 classified as +1 (Positive), rest of images classified as -1 (Negative).

**Testing Set:** 100 images: last 5 images from each series (20 series in all).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Maximum Margin Classifier.

**Parameters:**  $\nu = 0.01$ .

<b>Kernel</b>	<b>Classification</b>	<b># Testing</b>	<b># Correct</b>	<b># False</b>	<b># False</b>
<b>Width</b>	<b>Percentage</b>	<b>Points</b>		<b>Positives</b>	<b>Negatives</b>
3000	0.64	100	64	36	0
4000	0.80	100	80	20	0
6000	0.84	100	84	16	0

**Parameters:**  $\nu = 0.1$ .

<b>Kernel</b>	<b>Classification</b>	<b># Testing</b>	<b># Correct</b>	<b># False</b>	<b># False</b>
<b>Width</b>	<b>Percentage</b>	<b>Points</b>		<b>Positives</b>	<b>Negatives</b>
3000	0.64	100	64	36	0
4000	0.80	100	80	20	0
6000	0.84	100	84	16	0

**Notes:** There is no significant difference between  $\nu = 0.1$  and  $\nu = 0.01$ . The algorithm seems to find too many support vectors no matter what the kernel width is. 43 SVs for width = 3000 and 23 better SVs for width = 6000. This probably leads to a decision boundary that includes too many of the testing points. It makes sense since in all test cases we have only false positives. The tendency seems to be that the fewer support vectors, the better the classification, but even in the best scenario the classification percentage does not go above 84%.

**Experiment 10.** SVM Maximum Margin Classifier for Person 3.

**Purpose:** Identify pictures of Person 3 from a set of pictures of various people.

**Dataset:** First 200 images of database (5 people with 40 images each).

**Training Set:** 100 images: first 5 images from each series (20 series in all).

Person 3 classified as +1 (Positive), rest of images classified as -1 (Negative).

**Testing Set:** 100 images: last 5 images from each series (20 series in all).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Maximum Margin Classifier.

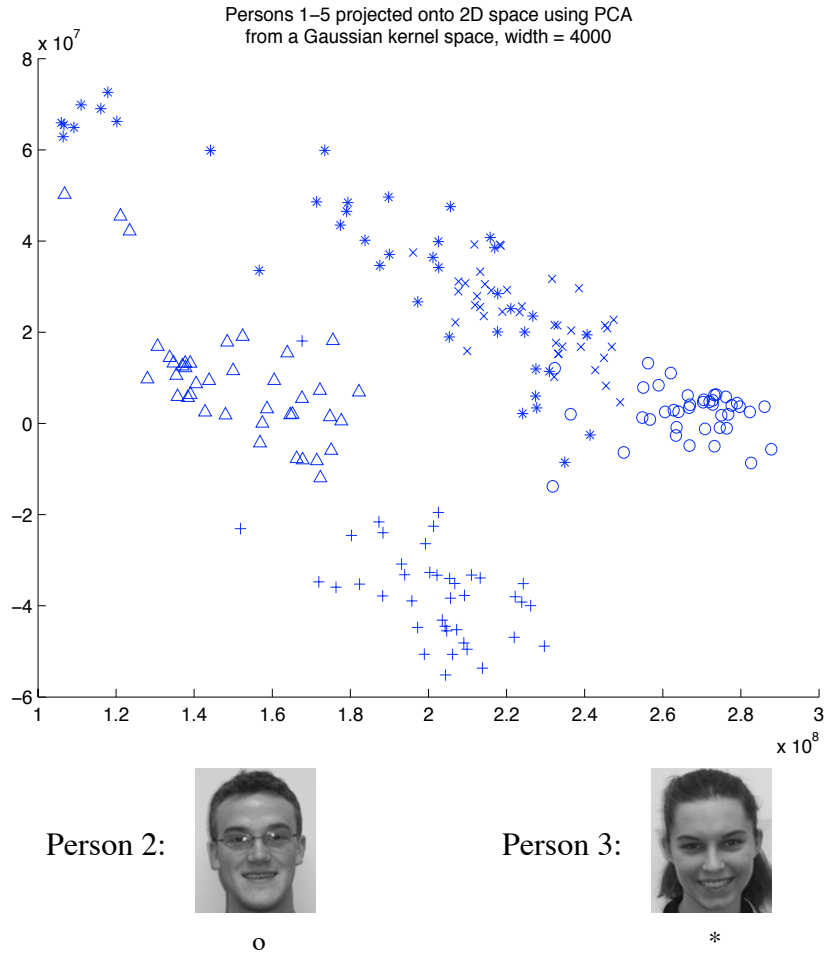
**Parameters:**  $\nu = 0.01$ .

<b>Kernel</b>	<b>Classification</b>	<b># Testing</b>	<b># Correct</b>	<b># False</b>	<b># False</b>
<b>Width</b>	<b>Percentage</b>	<b>Points</b>		<b>Positives</b>	<b>Negatives</b>
3000	0.50	100	50	50	0
4000	0.62	100	62	38	0
6000	0.70	100	70	30	0

**Parameters:**  $\nu = 0.1$ .

<b>Kernel</b>	<b>Classification</b>	<b># Testing</b>	<b># Correct</b>	<b># False</b>	<b># False</b>
<b>Width</b>	<b>Percentage</b>	<b>Points</b>		<b>Positives</b>	<b>Negatives</b>
3000	0.50	100	50	50	0
4000	0.62	100	62	38	0
6000	0.70	100	70	30	0

**Notes:** There is no significant difference between  $\nu = 0.1$  and  $\nu = 0.01$ . The algorithm seems to find too many support vectors no matter what the kernel width is. 55 SVs for width = 3000 and 35 better SVs for width = 6000. This probably leads to a decision boundary that includes too many of the testing points. It makes sense since in all test cases we have only false positives. The tendency seems to be that the less support vectors, the better the classification, but even in the best scenario the classification percentage does not improve beyond 70%. Comparing these results to Experiment 9, the errors are higher for Person 3 than for Person 2 because Person 3 had more varied pictures than Person 2 (see Figure 5.6). The algorithm makes the decision boundary larger in order to include all pictures of Person 3, but this leads to including more false positives as well.



**Figure 5.6. Person 3's vectors are more spread out than Person 2's vectors.**

## **Conclusions to Face Recognition Task:**

The Face Recognition task is straightforward for the more complex algorithms, such as Fisher Linear Discriminant Classifier, and SVM Novelty Detection. These algorithms are able to identify a person with a correct classification percentage as high as 98%. The best is the SVM Novelty Detection, which consistently classified more than 90% of the testing images when identifying both Person 2 and 3. The SVM Maximum Margin Classifier performs better than Naïve Novelty Detection and Parzan Window Classifier, but it does not reach the high performance of Fisher Linear Discriminant Classifier and SVM Novelty Detection. The first two algorithms in Experiments 1-4 clearly are not sufficient for this task, as they were not able to differentiate between different people.

In conclusion, the Face Recognition task is relatively easy.



# Chapter 6: Feature Identification— Hair Style










The goal of this task is to identify whether a person has long or short hair. One of the problems with this kind of classification is to objectively decide if a person has long hair or short hair. If one has medium length hair, then it is a matter of personal opinion whether one has short or long hair. Moreover, there are people where it is not clear from the picture if they have long or short hair. For example Person 4 has his hair pulled back in a long tail. However, it is not clear in every picture whether Person 4 has long hair or not.



Another problem is regarding the selection of good training and testing sets. The algorithms should learn if a person has long hair from only the picture of that person. We want the algorithms to avoid correlating the face of a person with the length of the hair in the training phase. In other words, the algorithms should learn the “hair length” feature, not the person with long or short hair. Thus we have one set of experiments (Experiments

11-15) that includes the same people with long hair in both the training and testing sets, and a second set of experiments (Experiments 16-20) where we use different people with long hair in the training and testing sets.

**Experiments 11-15:**

<p>First five images of these people classified as having long hair and included in <i>training</i>:</p>					
	Person 3	Person 8	Person 14	Person 16	Person 19
					
	Person 30	Person 31	Person 34	Person 39	

<p>Last five images of these people classified as having long hair and included in <i>testing</i>:</p>					
	Person 3	Person 8	Person 14	Person 16	Person 19
					
	Person 30	Person 31	Person 34	Person 39	

### Experiments 16-20:

All images of these people classified as having long hair and included in <i>training</i> :				
	Person 3	Person 14	Person 16	Person 39
	(Images 81-90)			

All images of these people classified as having long hair and included in <i>testing</i> :					
	Person 8	Person 19	Person 30	Person 31	Person 34

Within each set of five experiments, each experiment corresponds to one of the five algorithms we use: Naïve Novelty Detection, Parzan Window Classifier, Fisher Linear Discriminant Classifier, SVM Novelty Detection, and SVM Maximum Margin Classifier. We use a Gaussian kernel with widths of 3000, 4000 and 6000 throughout and a width of 9000 for particular experiments.

The format of each experiment is similar to the one in Chapters 5. For each experiment we provide a goal, the dataset used, the training set, the testing set, the type of kernel, the algorithm used, and the set of parameters for the algorithm. We present results for each set of parameters in a table specifying the kernel width used, the classification percentage, the number of testing points, the number of correct classifications, and a

break down of false positives and false negatives. We conclude each experiment with a discussion of the results.

We discuss major conclusions of the “long hair” feature identification task at the end of the chapter.

**Experiment 11.** Naïve Novelty Detection.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 45 images: first 5 images of Persons 3, 8, 14, 16, 19, 30, 31, 34, 39 (Negatives).

**Testing Set:** 545 images: other 5 images of people with long hair (Negatives) and all 500 images of people with short hair (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** Naïve Novelty Detection.

**Parameters:**  $\delta = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.0826	545	45	0	500
4000	0.0826	545	45	0	500
6000	0.0826	545	45	0	500

**Notes:** The algorithm considers all testing points non-novel, i.e. people with long hair. It appears that it learns a decision boundary that is too large and thus all testing points fall inside it.

**Experiment 12.** Parzan Window Classifier.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 295 images: first 5 images from each series (59 series in all). First 5 images of Person 3, and all images of Persons 8, 14, 16, 19, 30, 31, 34, 39 classified as +1 (Positives), rest of images classified as -1 (Negatives).

**Testing Set:** 295 images: last 5 images from each series (59 series in all).

**Kernel:** Gaussian kernel.

**Algorithm:** Parzan Window Classifier.

**Parameters:** none.

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.1525	295	45	250	0
4000	0.1525	295	45	250	0
6000	0.1525	295	45	250	0

**Notes:** The algorithm classifies all people as having long hair. Here we have a similar phenomenon as in Experiment 11.

**Experiment 13.** Fisher Linear Discriminant Classifier.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 295 images: first 5 images from each series (59 series in all). First 5 images of Person 3, and all images of Persons 8, 14, 16, 19, 30, 31, 34, 39 classified as +1 (Positives), rest of images classified as -1 (Negatives).

**Testing Set:** 295 images: last 5 images from each series (59 series in all).

**Kernel:** Gaussian kernel.

**Algorithm:** Fisher Linear Discriminant Classifier.





**Parameters:**  $\lambda = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.979	295	289	5	1
4000	0.983	295	290	4	1
6000	0.979	295	289	5	1

**Parameters:**  $\lambda = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.983	295	290	4	1
4000	0.986	295	291	3	1
6000	0.979	295	289	5	1

**Notes:**  $\lambda = 0.01$  provides slightly better results. For the Gaussian kernel width of 4000, the false positives are people who have a very different pose than their normal one, and the one false negative that appears in all testing cases is Person 3 who moved her hair behind her shoulders so less hair is visible. Otherwise the algorithm performs very well.

False Positives:	 Image 150	 Image 190	 Image 430	False Negatives:	 Image 88
---------------------	---	---	---	---------------------	--



**Experiment 14.** SVM Novelty Detection.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 45 images: first 5 images of Persons 3, 8, 14, 16, 19, 30, 31, 34, 39 (Negatives).

**Testing Set:** 545 images: other 5 images of people with long hair (Negatives) and all 500 images of people with short hair (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Novelty Detection.



**Parameters:**  $\gamma = 0.01, \nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.956	545	521	24	0
4000	0.968	545	528	17	0
6000	0.979	545	534	11	0
9000	0.983	545	536	9	0

**Parameters:**  $\gamma = 0.1, \nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.996	545	543	2	0
4000	0.996	545	543	2	0
6000	0.976	545	532	2	11
9000	0.882	545	481	0	64

**Notes:** With  $\gamma = 0.01, \nu = 0.1$ , the algorithm performs very well with a Gaussian kernel width of 9000. However, the best result is with  $\gamma = 0.1, \nu = 0.1$ , with the kernel width of 3000 or 4000; we have only two false positives and no false negatives. In image 229 we see that the person is positioned in such a way that a lot of his hair is not showing in the picture, while in image number 88 the person moved her hair partly behind her shoulders so most of it is not seen. These may be reasons as to why we get the two false positives.

False Positives:	 Image 88	 Image 229	False Negatives:	(none)
------------------	---	--	------------------	--------

**Experiment 15.** SVM Maximum Margin Classifier.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 295 images: first 5 images from each series (59 series in all). First 5 images of Person 3 and all images of Persons 8, 14, 16, 19, 30, 31, 34, 39 classified as +1 (Positives), rest of images classified as -1 (Negatives).

**Testing Set:** 295 images: last 5 images from each series (59 series in all).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Maximum Margin Classifier.

**Notes:** The algorithm did not finish solving the quadratic system of equations so we have no results.

**Experiment 16.** Naïve Novelty Detection.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 40 images: First 10 images of Person 3 and all images of Persons 14, 16, 39 (Negatives).

**Testing Set:** 550 images: all images of Persons 8, 19, 30, 31, 34 classified as having long hair (Negatives) and all other 500 images (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** Naïve Novelty Detection.

**Parameters:**  $\delta = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.09	550	50	0	500
4000	0.09	550	50	0	500
6000	0.09	550	50	0	500

**Notes:** The algorithm classifies every person as having long hair. The radius of the hyper-sphere is too large and it includes every testing image.

**Experiment 17.** Parzan Window Classifier.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 230 images: First 10 images of Person 3, and all images of Persons 2, 5, 14, 16, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44. Images of Persons 3, 14, 16, 39 classified as having long hair (Positives), rest of images classified as having short hair (Negatives).

**Testing Set:** 360 images: all images of Persons 8, 19, 30, 31, 34 classified as having long hair (Positives) and 310 images from other people classified as having short hair (Negatives).

**Kernel:** Gaussian kernel.

**Algorithm:** Parzan Window Classifier.

**Parameters:** none.

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.138	360	50	310	0
4000	0.138	360	50	310	0
6000	0.138	360	50	310	0

**Notes:** The algorithm is not able to learn a good decision boundary. It may be affected negatively by the large number of images with short hair in the training set.

**Experiment 18.** Fisher Linear Discriminant Classifier.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 230 images: First 10 images of Person 3, and all images of Persons 2, 5, 14, 16, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44. Images of Persons 3, 14, 16, 39 classified as having long hair (Positives), rest of images classified as having short hair (Negatives).

**Testing Set:** 360 images: all images of Persons 8, 19, 30, 31, 34 classified as having long hair (Positives) and 310 images from other people classified as having short hair (Negatives).

**Kernel:** Gaussian kernel.

**Algorithm:** Fisher Linear Discriminant Classifier.










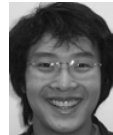
**Parameters:**  $\lambda = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.944	360	340	20	0
4000	0.947	360	341	19	0
6000	0.950	360	342	18	0

**Parameters:**  $\lambda = 0.01$ .









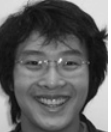
<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.958	360	345	15	0
4000	0.950	360	342	18	0
6000	0.936	360	337	23	0

**Notes:** When  $\lambda = 0.1$  and for all kernel widths, Person 23 (Images 371-380) is identified as having long hair, which makes sense since he does tend to have longer hair than Person 2 for example. So the classification for this person could be considered correct. The other pictures identified as having long hair are really false positives. These pictures include people with a hood behind their back (Images 254, 256, 260), Person 4 and Person 28 making a funny face (Images 150, 430), people facing different directions (Images 92, 417, 419) and people with more facial hair (Image 425). The algorithm misclassifies these extreme cases.

False Positives ( $\lambda = 0.1$ ):					
	Image 92	Image 150	Image 254	Image 256	Image 260
					
	Image 417	Image 419	Image 425	Image 430	Images 371-380

It is worth mentioning that we have no false negatives, so everybody with long hair is correctly detected. Also there are very few instances (Image 92) where Person 3 is detected as having long hair, just because her hair shows up from behind her back when her head is turned away. This suggests that the algorithm is not identifying people, but rather the "long hair" feature of people, since Person 3 was included in the training set with clearly visible long hair, but in the testing set she has her hair pulled back which is not clearly visible in most pictures. However, as soon as she turns her head and her pony tail becomes visible, the algorithm detects that she has long hair in those pictures.

When  $\lambda = 0.01$  the algorithm is more consistent, especially with a kernel width of 3000. The funny faces of Person 4 (Images 130, 150), the person with longer hair in image 311 (but only in this instance), as well as Person 11 with a hood behind his neck (Images 251-260) are all detected as having long hair. Person 23 has longer hair than most short hair people, so his images are false positives as well.

False Positives ( $\lambda = 0.01$ ):					
	Image 130	Image 150	Image 252	Image 254	Image 256
					
	Image 259	Image 260	Image 311	Images 371-380 (Person 23)	



It is clear that the algorithm detects long hair behind people's necks and above the shoulders. If the hair is pulled back and not directly visible in the picture, or if it is not reaching the shoulders of the person but it is medium in length, then the algorithm has trouble identifying long hair versus short hair. Also, some people would disagree whether medium length hair is short or long. The kind of classification that the algorithm makes can be considered its own personal "opinion." Thus, it is difficult to evaluate the actual performance percentage since humans may not agree with a certain classification.

**Experiment 19.** SVM Novelty Detection.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 40 images: First 10 images of Person 3 and all images of Persons 14, 16, 39 (Negatives).

**Testing Set:** 550 images: all images of Persons 8, 19, 30, 31, 34 classified as having long hair (Negatives) and all other 500 images (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Novelty Detection.

**Parameters:**  $\gamma = 0.01, \nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.915	550	503	47	0
4000	0.916	550	504	46	0
6000	0.920	550	506	44	0
9000	0.924	550	508	42	0

**Parameters:**  $\gamma = 0.1, \nu = 0.1$ .















<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.936	550	515	35	0
4000	0.949	550	522	28	0
6000	0.955	550	525	22	3
9000	0.955	550	525	11	14

**Notes:** When  $\gamma = 0.01, \nu = 0.1$  even if the classification percentage is high, the algorithm misclassifies most of the people with long hair (there are 50 pictures of people with long hair in the testing set and more than 40 pictures get misclassified). It is clearly not learning the “long hair” feature.












When  $\gamma = 0.1, \nu = 0.1$  the algorithm performs a little better and for a kernel width of 9000 it is actually learning the "long hair" feature. We have several false negatives meaning the algorithm considered these people to have long hair, although they cannot be clearly classified as having long or short hair.

For the false negatives the people are debatable whether they have short or long hair. Images 256 and 260 show a person with longer hair and much facial hair. Images 291, 298, 299, and 300 are a long hair person who has her hair pulled back but some of it is hanging on the sides of her face so it is more visible. Images 322 and 327 are Person 18 who has medium length hair. It is interesting that not all Person 18’s pictures are detected as having long hair. It is probably because his hair seems a little longer in these two pictures and also because he is wearing a hood behind his neck which affects the

algorithm's classification. Images 91, 100, 101, 109, 111 are Person 3 where we see some of her long hair but not all of it. Note that this algorithm detects the "long hair" feature in a different way than the Fisher Linear Discriminant Classifier in Experiment 18 does where Person 3 needed to be turned so that her pony tail would show up. Image 160 is Person 4 with his shoulders raised a lot, which probably affected the algorithm and classified the person as having long hair in this picture.

False							
	Image 91	Image 100	Image 101	Image 109	Image 111	Image 160	Image 256
Negatives:							
	Image 260	Image 291	Image 298	Image 299	Image 300	Image 322	Image 327

Note that even most of the false positives can be explained. Person 8 (Images 221-230) has medium length hair, and the algorithm classified him as having short hair. If we do not consider Person 8's misclassification, we actually get a classification performance of  $535 / 550 = 0.973$  which is better than the Fisher Linear Discriminant Classifier in Experiment 18. This performance is very good considering that this algorithm is a just Novelty Detector and that the training set contains only 40 points compared to the 550 testing points.

False Positives:						
	Image 221	Image 222	Image 223	Image 224	Image 225	Image 226
						
	Image 227	Image 228	Image 229	Image 230	Image 338	

In light of these comments the algorithm seems to perform very well. Since no person can make a clear cut binary decision regarding the hair length of certain people, we can just give credit to the algorithm for considering a medium length hair person (such as Person 8) as having short hair. This decision could be interpreted as just the algorithm's "personal opinion."

**Experiment 20.** SVM Maximum Margin Classifier.

**Purpose:** Identify people with long hair from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** 230 images: First 10 images of Person 3, and all images of Persons 2, 5, 14, 16, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44. Images of Persons 3, 14, 16, 39 classified as having long hair (Positives), rest of images classified as having short hair (Negatives).

**Testing Set:** 360 images: all images of Persons 8, 19, 30, 31, 34 classified as having long hair (Positives) and 310 images from other people classified as having short hair (Negatives).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Maximum Margin Classifier.

**Parameters:**  $\nu = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.397	360	143	217	0
4000	0.519	360	187	173	0
6000	0.775	360	279	81	0

**Note:** When kernel width = 6000, the algorithm did not finish solving the quadratic equations.

**Parameters:**  $\nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.402	360	145	215	0
4000	0.500	360	180	180	0
6000	0.911	360	328	32	0








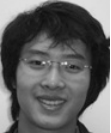


**Note:** When kernel width = 6000, the algorithm did not finish solving the quadratic equations.

**Notes:** When  $\nu = 0.01$  and kernel width is 6000 the algorithm has 30 support vectors and, although it seems that it has a higher classification error than Fisher Linear Discriminant Classifier (Experiment 18) or the SVM Novelty Detection (Experiment 19), most of the misclassified points are people with medium length hair or the length of their hair is not clearly visible from the picture.

The algorithm detects the people with long hair perfectly, while it has some "doubts" about 81 images. We do not provide all false positives for brevity, but we do include images of the people who are misclassified. Following is a discussion of some of the misclassified images.

Persons 7, 11, 15, 17 and 23 have medium length hair and are mostly classified as having long hair. Person 18 has medium length hair and is mostly classified as having short hair. Person 24 clearly has short hair, but is mostly classified as having long hair. We hypothesize the misclassification occurs because of the dark jacket that appears in the pictures. Persons 3 and 4 are inconsistently classified as having either long or short hair,

probably because their long hair is not clearly visible in all pictures. Person 25 is consistently classified as having short hair with the exception of his last image where his hands probably affect the algorithm.

False Positives:					
	Person 3	Person 4	Person 7	Person 11	Person 15
					
	Person 17	Person 18	Person 23	Person 24	Person 25

Generally, it is difficult to assess what affects the algorithm, but it is clear that when a person has more hair showing in the picture, the algorithm rates the person as having long hair correctly. Note again that Person 3 was used in the training set with clearly long hair showing in the pictures, and since there are more pictures of her from the testing set classified as having short hair, it means the algorithm is really learning the "long hair" feature and not the face of the person.

After analyzing many of the false positives we realize that the algorithm just seems to have a different "opinion" about what long hair is compared to other algorithms. However, it is still classifying certain people with short hair the wrong way (Person 24). Although with some people such as Person 11 the algorithm is consistent, in other cases, such as Person 4, it is clearly inconsistent.

When  $\nu = 0.1$ , the algorithm is even more inconsistent, even if the classification percentage is higher than when  $\nu = 0.01$ . Person 24 is correctly classified as having short



hair, but the other false positives are the same as when  $\nu = 0.01$  with even more inconsistencies throughout.

In conclusion, it is difficult to assess the performance of the algorithm in the two cases of  $\nu = 0.1$  and  $\nu = 0.01$ .

## **Conclusions to Hair Style Feature Identification Task:**

The Naïve Novelty Detection and Parzan Window Classifier algorithms do not work for this task. The other algorithms perform very well in all experiments.

For the algorithms that work, experiments 11-15 show that they are able to learn if a person has long hair and almost consistently classify other pictures of the same person correctly. In this set of experiments we have higher classification percentages than in the second set of experiments probably because it is easier to identify a similar picture of the same person and classify that person as having long or short hair.

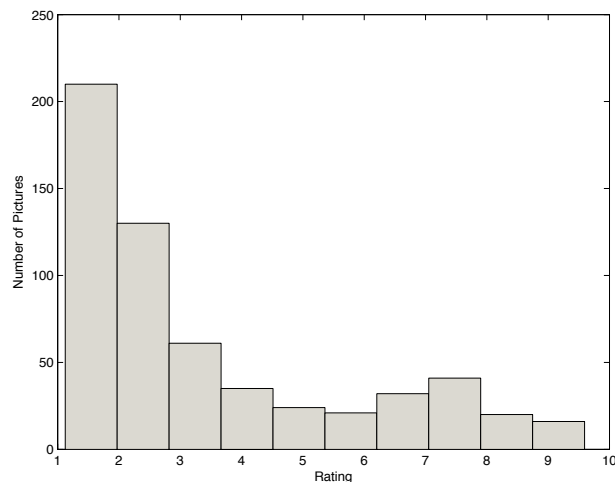
The second set of experiments (16-20) proves that the algorithms learn the “long hair” feature of the people in the images, not an association of the “long hair” label with a particular person, as is the case in the first set of experiments. This is supported by the observation that Person 3 is classified as having short hair in several instances of the testing images, although pictures of her showing long hair are used as training images. Moreover, the algorithms deal with people having medium length hair in a remarkable way. Although there are varying degrees of inconsistencies among them, the algorithms usually tend to choose one classification over the other for the majority of pictures of one particular person.

This task seems relatively easy for the algorithms, although slightly more difficult than Face Recognition. One of the difficulties of evaluating the task and the performance of the algorithms rises from the possible disagreement between humans over the classification of medium length hair.

# Chapter 7: Feature Identification— Smiling

The last task in our paper is learning to identify smiling people from a set of pictures with people showing different moods.

Using the ratings of the photos we divide the database into two datasets in order to see how the algorithms perform on the different cases. Recall the histogram of the smile ratings in Figure 7.1.



*Figure 7.1. Histogram of smile ratings.*

When the cutoff point is 6.0, the first dataset contains 113 people with smile ratings of 6.0 and higher and 477 people with smile ratings below 6.0. A cutoff point of 8.0 specifies the second dataset with 31 people with smile ratings of 8.0 and higher and 559 people with ratings below 8.0. During the experiments we consider people with ratings equal to or above the cutoff point as smiling, while people with ratings below the cutoff point as non-smiling. We divide the smiling people in half for training and use the other half for testing for each dataset.

There are ten experiments in total, five for the first dataset and five for the second dataset. Within a set of five experiments, each corresponds to one of the five algorithms we use: Naïve Novelty Detection, Parzan Window Classifier, Fisher Linear Discriminant Classifier, SVM Novelty Detection, and SVM Maximum Margin Classifier. We use a Gaussian kernel with widths of 3000, 4000 and 6000.

The format of each experiment is similar to the one in Chapters 5 and 6. For each experiment we provide a goal, the dataset used, the training set, the testing set, the type of kernel, the algorithm used, and the set of parameters for the algorithm. We present results for each set of parameters in a table specifying the kernel width used, the classification percentage, the number of testing points, the number of correct classifications, and a break down of false positives and false negatives. We conclude each experiment with a discussion of the results.

We discuss major conclusions of the smile identification task at the end of the chapter.

**Experiment 21.** Naïve Novelty Detection—cutoff 6.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 56 smiling images (Negatives).

**Testing Set:** The other 57 smiling image (Negatives), and 477 non-smiling images (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** Naïve Novelty Detection.

**Parameters:**  $\delta = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.1067	534	57	0	477
4000	0.1067	534	57	0	477
6000	0.1067	534	57	0	477

**Notes:** The algorithm is classifying all pictures as smiling. We assume that the very high dimensionality of the input space is affecting the algorithm in the training part, so it forms a large radius that contains all training and testing points.

**Experiment 22.** Naïve Novelty Detection—cutoff 8.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 15 smiling images (Negatives).

**Testing Set:** The other 16 smiling image (Negatives), and 559 non-smiling images (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** Naïve Novelty Detection.

**Parameters:**  $\delta = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.0278	575	16	0	559
4000	0.0278	575	16	0	559
6000	0.0278	575	16	0	559

**Notes:** (Similar conclusion to Experiment 21). The algorithm is classifying all pictures as smiling. We assume that the very high dimensionality of the input space is affecting the algorithm in the training part, so it forms a large radius that contains all training and testing points.

**Experiment 23.** Parzan Window Classifier—cutoff 6.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 56 smiling images (Positives) and first 238 non-smiling images (Negatives).

**Testing Set:** The other 57 smiling image (Positives), and the other 239 non-smiling images (Negatives).

**Kernel:** Gaussian kernel.

**Algorithm:** Parzan Window Classifier.

**Parameters:** none.

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.1926	296	57	239	0
4000	0.1926	296	57	239	0
6000	0.1926	296	57	239	0

**Notes:** The algorithm is classifying all pictures as smiling. We have a similar phenomenon as in Experiments 21 and 22.

**Experiment 24.** Parzan Window Classifier—cutoff 8.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 15 smiling images (Positives) and first 279 non-smiling images (Negatives).

**Testing Set:** The other 16 smiling image (Positives), and the other 280 non-smiling images (Negatives).

**Algorithm:** Parzan Window Classifier.

**Parameters:** none.

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.0541	296	16	280	0
4000	0.0541	296	16	280	0
6000	0.0541	296	16	280	0

**Notes:** The algorithm is classifying all pictures as smiling. We have a similar phenomenon as in Experiments 21, 22 and 23.



**Experiment 25.** Fisher Linear Discriminant Classifier—cutoff 6.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 56 smiling images (Positives) and first 238 non-smiling images (Negatives).

**Testing Set:** The other 57 smiling image (Positives), and the other 239 non-smiling images (Negatives).

**Kernel:** Gaussian kernel.

**Algorithm:** Fisher Linear Discriminant Classifier.

**Parameters:**  $\lambda = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.5169	296	153	119	24
4000	0.5845	296	173	95	28
6000	0.6385	296	189	74	33

**Notes:** Although it is difficult to evaluate what causes such a poor performance, the algorithm is able to detect roughly half of the smiles and more than half of the non-smiles correctly.

**Experiment 26.** Fisher Linear Discriminant Classifier—cutoff 8.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 15 smiling images (Positives) and first 279 non-smiling images (Negatives).

**Testing Set:** The other 16 smiling image (Positives), and the other 280 non-smiling images (Negatives).

**Kernel:** Gaussian kernel.

**Algorithm:** Fisher Linear Discriminant Classifier.

**Parameters:**  $\lambda = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.5000	296	148	140	8
4000	0.5777	296	171	117	8
6000	0.6655	296	197	91	8

**Notes:** Although it is difficult to evaluate what causes such a poor performance, the algorithm is able to detect half of the smiles and more than half of the non-smiles correctly. Note that the classification percentage is similar to Experiment 25 even if the cutoff is higher.

**Experiment 27.** SVM Novelty Detection—cutoff 6.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 56 smiling images (Negatives).

**Testing Set:** The other 57 smiling image (Negatives), and 477 non-smiling images (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Novelty Detection.

**Parameters:**  $\gamma = 0.1, \nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.1498	534	80	3	451
4000	0.1479	534	79	4	451
6000	0.1199	534	64	2	468

**Parameters:**  $\gamma = 0.01, \nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.6685	534	357	32	145
4000	0.4120	534	220	20	294
6000	0.2903	534	155	14	365

**Notes:** This algorithm has the same problems as the Naïve Novelty Detection in Experiment 21. The spread of smiling images among non-smiling images and the lack of linearly separable data in the feature space are likely causes for the low performance. Decreasing  $\gamma$  to 0.01, thus making the decision boundary tighter seems to support this theory since results improve. Although more than a half of the non-smiling pictures are detected correctly, less than half of the smiling pictures are detected correctly.

**Experiment 28.** SVM Novelty Detection—cutoff 8.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 15 smiling images (Negatives).

**Testing Set:** The other 16 smiling image (Negatives), and 559 non-smiling images (Positives).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Novelty Detection.

**Parameters:**  $\gamma = 0.01, \nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.9600	575	552	16	7
4000	0.8330	575	479	12	84
6000	0.5478	575	315	8	252

**Notes:** The high classification percentage with a kernel width of 3000 is misleading. The algorithm has trouble classifying all smiling images correctly. If we increase the kernel width up to 6000, the algorithm is able to classify half of the smiling images correctly, but it misclassifies almost half of the non-smiling images. Although the algorithm performs better with a higher cutoff point than in Experiment 27, its limitations are similar. This is an example of how difficult the smile classification task is.

**Experiment 29.** SVM Maximum Margin Classifier—cutoff 6.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 56 smiling images (Positives) and first 238 non-smiling images (Negatives).

**Testing Set:** The other 57 smiling image (Positives), and the other 239 non-smiling images (Negatives).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Maximum Margin Classifier.

**Parameters:**  $\nu = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.4291	296	127	166	3
4000	0.5203	296	154	129	13
6000	0.6014	296	178	105	13

**Parameters:**  $\nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.4291	296	127	166	3
4000	0.5203	296	154	129	13
6000	0.5304	296	157	127	12

**Notes:** There is no significant difference between  $\nu = 0.01$  and  $\nu = 0.1$ .

Generally the algorithm cannot easily differentiate between smiling and non-smiling images. However, with a kernel width of 6000 and  $\nu = 0.01$ , the algorithm performs the best. The misclassifications are well balanced. More than half of the non-smiling images and 78% of the smiling images are classified correctly. This algorithm seems to classify smiling images better than the SVM Novelty Detection in Experiment 27.

**Experiment 30.** SVM Maximum Margin Classifier—cutoff 8.0.

**Purpose:** Identify smiling people from a set of pictures of various people.

**Dataset:** All 590 images.

**Training Set:** First 15 smiling images (Positives) and first 279 non-smiling images (Negatives).

**Testing Set:** The other 16 smiling image (Positives), and the other 280 non-smiling images (Negatives).

**Kernel:** Gaussian kernel.

**Algorithm:** SVM Maximum Margin Classifier.

**Parameters:**  $\nu = 0.01$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.5777	296	171	117	8
4000	0.6622	296	196	92	8
6000	0.8176	296	242	45	9

**Note:** When kernel width = 6000, the algorithm did not finish solving the quadratic equations.

**Parameters:**  $\nu = 0.1$ .

<b>Kernel Width</b>	<b>Classification Percentage</b>	<b># Testing Points</b>	<b># Correct</b>	<b># False Positives</b>	<b># False Negatives</b>
3000	0.5608	296	166	123	7
4000	0.6858	296	203	85	8
6000	0.7162	296	212	76	8



**Note:** When kernel width = 4000 and 6000, the algorithm did not finish solving the quadratic equations.

**Notes:**  $\nu = 0.01$  is a better choice than  $\nu = 0.1$  considering peak performance of the algorithm. With  $\nu = 0.01$  the algorithm is able to classify non-smiling images better than the SVM Novelty Detection in Experiment 28, probably because it correlates the training images with the classification. Moreover, there are only half misclassified smiling images. Even if the algorithm did not completely solve the quadratic equations with a kernel width of 6000, it still has a high correct classification percentage.

## **Conclusions to Smile Feature Identification Task:**

The Naïve Novelty Detection and Parzan Window Classifier algorithms do not work for this task. The other algorithms perform better, but with clear limitations. Most of the classification percentages of latter algorithms lie within the 50%-70% range. Although they are able to classify correctly some of the images, generally more than half of either smiling or non-smiling images are classified incorrectly. On closer inspection, classification percentages above 90% are misleading. In these cases the algorithms tend to classify most pictures as non-smiling and the small number of smiling pictures in the testing set makes the classification percentage high, even if most of the smiling pictures are misclassified.

These problems hint at the difficulty of the smile feature identification task.

# Chapter 8:

## Conclusions

We are able to evaluate the performance of the five algorithms we used for our tasks. Both Naïve Novelty Detection and Parzan Window Classifier are not sufficient to learn any classification in any of the experiments. We hypothesize that the high number of dimensions and the complicated nature of the Gaussian kernel defined feature space were the causes of their failure.

The Fisher Linear Discriminant Classifier performs consistently well throughout. The Face Recognition Task and the Hair Style Feature Identification Task are easy to solve using this algorithm, but Smile Feature Identification proves to be slightly more difficult. Fisher Linear Discriminant Classifier is obviously better than Naïve Novelty Detection and Parzan Window Classifier. First, Fisher Linear Discriminant Classifier correlates the training labels with the training points to improve classification percentages over Naïve Novelty Detection. Second, even if Parzan Window Classifier does the same, recall that Fisher Linear Discriminant Classifier takes into account the number of training points as well as the density of each class of points to learn a better decision boundary.

SVM Novelty Detection is slightly better than Fisher Linear Discriminant Classifier, and in several instances the advantages of using support vectors help this

algorithm achieve even better results. For this algorithm, Face Recognition is an easy task as well as classifying people as having long or short hair. Smile Feature Identification is more difficult, but generally the algorithm performs slightly better at this task than Fisher Linear Discriminant Classifier.

SVM Maximum Margin Classifier proves to be the most inconsistent algorithm across all tasks. Although Face Recognition is mostly straightforward for other algorithms, SVM Maximum Margin Classifier's performance is average with only 80%-84% of the images being identified correctly. In the case of Hair Style Feature Identification, the algorithm is not as good as SVM Novelty Detection, but reaches an arguably good classification percentage. For Smile Feature Identification the algorithm is the best, since it classifies both smiling and non-smiling people with a balanced accuracy.

Regarding the classification tasks, Face Recognition is straightforward and Hair Style Feature Identification is slightly more difficult but still relatively easy. Smile Feature Identification on the other hand proves to be very difficult for all five algorithms.

It is worth mentioning that the algorithms do not necessarily fail classifying people as smiling or not, but that the high dimensionality of the input space together with the use of the Gaussian kernel determines the misclassification. In other words, for a difficult task such as this one, lower dimensionality feature spaces need to be hand crafted for the algorithm to look at only key areas of the photographs. Thus, for future research, we may try more advanced preprocessing of the images such as image segmentation and more sophisticated normalization [3, 4].

# Acknowledgments

I would like to thank my research supervisor and academic advisor, Dr. M. Kretchmar, for all the work he put in this project.

I would also like to thank David Nassar, Nathan Schmidt, Pam Arbisi, Tun Nyein, Jon Efe, Todd Feil, Pancham Gajjar, Josh Mankoff, Emily Hampp, Michael Williams, Mete Tuzcu, Matthew St. John, Thomas Bressoud, Adam Hitchcock, Tony Silveira, John Szendiuch, Andrew Hoffman, Lewis Ludwig, Matthew Neal, Patrick Commins, Colleen Hughes, Robey Holderith, Samuel Behrend, Michael Bono, Jonathan Pierce, Barry Westmoreland, Max Quinlin, Chengeng Zeng, Kabir Mehra, Chinmoy Bhatiya, Scott Ratermann, Rachel Spence, Michael Palzkill, Charlie Fear, Ian Hudson, Anne Young, Nicole Scholtz, Robert Christian, Nathan Payne, Alexander Acheson, Jeffrey Camealy and the ones that I missed for participating in the development of the database.

Last, but not least, I would like to thank the 57 people who rated all 590 images for smiling content.



# Bibliography

- [1] Bartlett, Marian Stewart, Javier R. Movellan, and Terrence J. Sejnowski. *Face Recognition by Independent Component Analysis*. IEEE Transactions on Neural Networks. Vol 13. No 6. November 2002.
- [2] Boser, B. E., I. M. Guyon, and V. Vapnik. *A training algorithm for optimal margin classifiers*. Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory. Pittsburgh: ACM Press. 1992.
- [3] Delac, Kresimir, and Mislav Grgic. <http://www.face-rec.org/>
- [4] Delac, K., M. Grgic, and S. Grgic. *A Comparative Study of PCA, ICA, and LDA*. Proceedings of the 5th EURASIP Conference focuses on Speech and Image Processing, Multimedia Communications and Services, EC-SIP-M 2005.
- [5] Duda, Richard O., Peter E. Hart, and David G. Stork. Pattern Classification. 2nd Ed. New York, NY: John Wiley & Sons, Inc. 2001.
- [6] Hassoun, Mohamad H. Fundamentals of Artificial Neural Networks. Cambridge, MA: The MIT Press. 1995.
- [7] Haykin, Simon. Neural Networks: A Comprehensive Foundation. Upper Saddle River, NJ: Prentice Hall. 1999.
- [8] Kirby, Michael. Geometric Data Analysis. New York, NY: John Wiley & Sons, Inc. 2001.
- [9] Kohonen, Teuvo. Self Organizing Maps. Heidelberg, Germany: Springer-Verlag. 1995.
- [10] Lu, Juwei, Kostantinos N. Plataniotis, and Anastasios N. Venetsanopoulos. *Face Recognition Using LDA-Based Algorithms*. IEEE Transactions on Neural Networks. Vol 14. No 1. January 2003.

- [11] Lu, Juwei, Kostantinos N. Plataniotis, and Anastasios N. Venetsanopoulos. *Face Recognition Using Kernel Direct Discriminant Analysis Algorithms*. IEEE Transactions on Neural Networks. Vol 14. No 1. January 2003.
- [12] Martinez, Aleix M. and Avinash C. Kak. *PCA versus LDA*. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol 23. No 2. 2001.
- [13] Parzan, E. *Extraction and detection problems and reproducing kernel Hilbert spaces*. Journal of the Society for Industrial and Applied Mathematics. Series A. on control. 1962.
- [14] Shawe-Taylor, John, and Nello Cristianini. Kernel Methods for Pattern Analysis. Cambridge, UK: Cambridge University Press. 2004.
- [15] Schölkopf, Bernhard. Support Vector Learning. Munich: Oldenbourg Verlag. 1997.
- [16] Schölkopf, Bernhard, and Alexander J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. Cambridge, MA: The MIT Press. 2002.
- [17] Schölkopf, Bernhard, Alexander J. Smola, and Klaus-Robert Müller. *Nonlinear Component Analysis as a Kernel Eigenvalue Problem*. Max-Planck Institute for Biological Research. Technical Report No 44. December, 1996.
- [18] Sutton, Richard S., and Andrew G. Barto. Reinforcement Learning: An Introduction. Cambridge, MA: The MIT Press. 1998.



# Appendix A: Images

This appendix contains all 590 images that compose our database.

Person1: Images 1-10



Person 1: Images 11-20



Person 1: Images 21-30



Person 1: Images 31-40



Person 2: Images 41-50



Person 2: Images 51-60



Person 2: Images 61-70



Person 2: Images 71-80



Person 3: Images 81-90



Person 3: Images 91-100



Person 3: Images 101-110



Person 3: Images 111-120



Person 4: Images 121-130



Person 4: Images 131-140



Person 4: Images 141-150



Person 4: Images 151-160



Person 5: Images 161-170



Person 5: Images 171-180



Person 5: Images 181-190



Person 5: Images 191-200



Person 6: Images 201-210



Person 7: Images 211-220



Person 8: Images 221-230



Person 9: Images 231-240



Person 10: Images 241-250



Person 11: Images 251-260



Person 12: Images 261-270



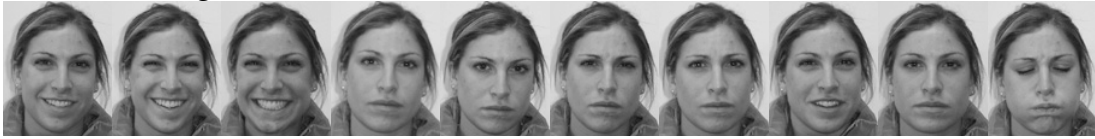
Person 13: Images 271-280



Person 14: Images 281-290



Person 15: Images 291-300



Person 16: Images 301-310



Person 17: Images 311-320



Person 18: Images 321-330



Person 19: Images 331-340



Person 20: Images 341-350



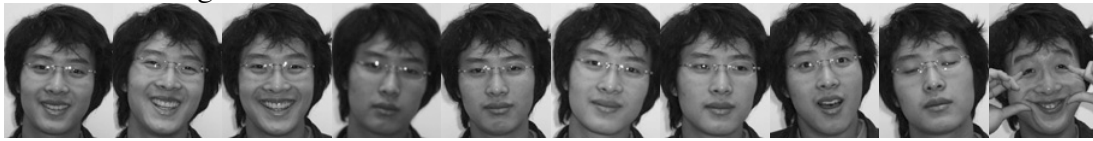
Person 21: Images 351-360



Person 22: Images 361-370



Person 23: Images 371-380



Person 24: Images 381-390



Person 25: Images 391-400



Person 26: Images 401-410



Person 27: Images 411-420



Person 28: Images 421-430



Person 29: Images 431-440

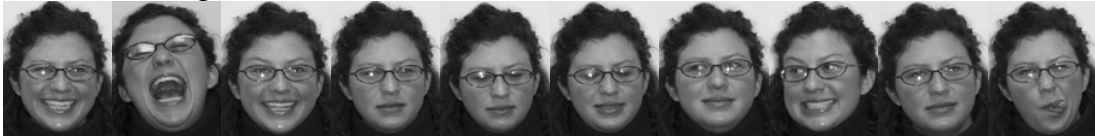


Person 30: Images 441-450





Person 31: Images 451-460



Person 32: Images 461-470



Person 33: Images 471-480



Person 34: Images 481-490



Person 35: Images 491-500



Person 36: Images 501-510



Person 37: Images 511-520



Person 38: Images 521-530



Person 39: Images 531-540



Person 40: Images 541-550



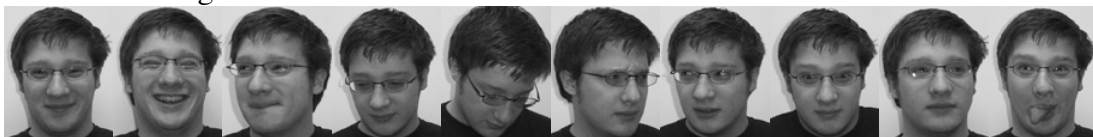
Person 41: Images 551-560



Person 42: Images 561-570



Person 43: Images 571-580



Person 44: Images 581-590

