|  |
| --- |
| **SVM Project and Gender Classification** |
| **Josh Gleason and Rod Pickens** |
| **Programming Assignment 4** |
| **Computer Science 679 – Pattern Recognition, UNR, Dr. Bebis** |
| **Due: Monday May 4, 2015** |
| **Submitted: Monday Monday 4, 2015** |

Contents

[1 Abstract 4](#_Toc417749430)

[2 Technical Discussion 4](#_Toc417749431)

[2.1 Gender classification 5](#_Toc417749432)

[2.1.1 Eigenvectors 5](#_Toc417749433)

[2.1.2 Eigenfaces 5](#_Toc417749434)

[2.2 Machine Learning 6](#_Toc417749435)

[2.2.1 Ideal Learning 8](#_Toc417749436)

[2.2.2 Inductive Learning 8](#_Toc417749437)

[2.2.3 Empirical Risk Minimization 10](#_Toc417749438)

[2.2.4 Structural Risk Minimization 11](#_Toc417749439)

[2.2.5 Penalization 13](#_Toc417749440)

[2.2.6 Minimum Description Length 13](#_Toc417749441)

[2.2.7 Bayes Induction 13](#_Toc417749442)

[2.3 Support Vector Machines 14](#_Toc417749443)

[2.4 Kernel Functions 17](#_Toc417749444)

[3 Project 17](#_Toc417749445)

[3.1 Experiment 1: SVM 17](#_Toc417749446)

[3.1.1 16x20 Images 17](#_Toc417749447)

[3.1.2 48x60 Images 18](#_Toc417749448)

[3.2 Experiment 2: Bayesian 18](#_Toc417749449)

[3.2.1 Training Parameters 18](#_Toc417749450)

[3.2.2 Testing on 16x20 Images 18](#_Toc417749451)

[3.2.3 Testing on 48x60 Images 18](#_Toc417749452)

[4 Conclusion 18](#_Toc417749453)

[4.1 Part A: SVM 18](#_Toc417749454)

[4.2 Part B: Bayesian 18](#_Toc417749455)

[4.3 Comparison 18](#_Toc417749456)

[5 Contributors 18](#_Toc417749457)

# Abstract

This paper describes our research regarding the fourth class project for the Computer Science pattern recognition class CS 679 taught by Dr. Bebis who is Department Chair of the Computer Science Department at the University of Nevada in Reno, Nevada.

The primary topics included in the main body of this report are a description of the project, learning machines, induction theory, theory of support vector machines (SVM), theory of kernel functions, and classifier results. The paper also compares SVM with Bayes classifier.

We added a section to our project report concerning the general topic of ***induction theory of learning***, and we did this for the purpose of rounding our understanding of learning machines. We did this because the class lecture on SVMs introduced us to two inductive principles used in machine learning: in particular, empirical risk minimization versus structured risk minimization.[[1]](#footnote-1)

In researching these two terms (empirical risk minimization and structured risk minimization), we were amazed to learn about the abstraction in machine learning based on the inductive principle: as we understand, machine learning is based on selecting an induction principle and then upon selecting a learning method. Given this new view, we definitely appreciate the class lecture on empirical risk minimization learning versus structured risk minimization learning. Because of that lecture, we did outside research and learned a whole new abstraction to the task of machine learning. We will discuss what we have learned (maybe mislearned) in a tutorial discussion below on inductive principles and their associated learning methods.

# Technical Discussion

This project depends upon gender classification using as features eigenfaces which, in this project, are non-linearly mapped to higher dimensional abstract feature space where a hyper-plane is used as the class discriminant. The eigenfaces and associated classification theory were studied in the previous project and learned in the class on linear discriminant function. The theory of eigenfaces are discussed beginning in Section 2.1.

The theory for support vector machines is related to the general theory of inductive methods, and from this general theory, we derive support vector machines from the inductive method class known as structured risk minimization. The discussion of inductive methods begins in section 2.3 and concludes with the theory of support vector machines.

## Gender classification

### Eigenvectors

Eigenvectors are those vectors that are invariant in direction to the action of a matrix A on the vector as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

In the above, is a square nxn matrix, is an nx1 vector, and is a scalar. As mentioned above, the action of the matrix upon the vector is to scale, and to only scale, the vector and not to change the direction of the vector

### Eigenfaces

The eigenface approach taken and experimented in this paper is from the research of Turk and Pentland,[[2]](#footnote-2) and their work was motivated by the earlier works of Sirovich and Kirby who represented pictures using principal component analysis.[[3]](#footnote-3) Principal component analysis is a least squares approach for minimizing the error associated with a projection of the data onto a different basis.

Given a vector in an N dimensional space, it can be represented by a set of N orthogonal basis vectors as follows

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

The goal is to find an N x K transformation matrix U such that

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

Where is an Nx1 rasterized image

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

And the basis vectors the right side of the equation are the standard basis (natural or canonical) vectors of Euclidean space as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

Where is known as the Kronecker delta function

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Yielding a set of basis vectors as follows

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

Moreover, the vector can be represented by a set of K orthogonal basis vectors *i*=1,…,K in a lower K (K<N) dimensional space, and each vector is an Nx1 vector. The lower K, again where K<N, dimensional space is defined as

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Principal component analysis selects the basis and coefficients to minimize the error

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

## Machine Learning

Machine learning will be described using the same formalism as Cherkassky in that he specifics the learning as taking the input from a generator (x) and a system (y), and from these inputs generating a learning machine that takes the inputs x and y and produces an approximation to y given we have no information on the true value of y, Figure 1.



Figure 1: Learning Machine (Cherkassky[[4]](#footnote-4))

The learning machine is based on a function f(x,w\*) that approximates the output from both the generator function g(x,w0) and the system function h(x,w0). The function f(x,w\*) is estimated from training data, and the purpose of f(x,y,w\*) is to approximate the output of the system h(x,w0):

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

The input x and y to the learning machine are independent and identically distributed (i.i.d.) pairs of data (*x,y*) distributed according to an unknown probability distribution function

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

If the test samples have the same probability distribution as the training samples, then the learning machine should correctly label unknown input samples x with the appropriate label y.

The best function[[5]](#footnote-5) for learning should be that function that minimizes the expected risk (error)

|  |  |  |
| --- | --- | --- |
|  |  | (12) |
|  |  |  |
|  |  |  |

The function is the loss function and P(x,y) is the known joint probability density function of the input x and y. Assuming a 0/1 loss function, then the expected risk is the minimum error function.

### Ideal Learning

The ideal learning machine is one such that the researcher is completely aware of the model of the probability distribution for the data x, the parameters describing the distribution, the system function h(x,w), and the cost function. With this a-priori knowledge, then a research can design a strong learning machine. The researcher will based the learning method upon that set of rules that minimize the expected risk.

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

Operating under these learning conditions will produce an optimal learning machine.

### Inductive Learning

When the conditions for ideal learning are unavailable, e.g. the probability distribution is unknown, then a researcher will be required to select from one of a number of inductive learning methods to base his machine learning engine. Once an inductive principle is selected, then the researcher will have to select a particular learning method to construct the learning machine.

When the probability distributions are unknown and the expected risk cannot be determined, researchers often resort to designing their learning machine using an induction principle. A simple induction principle, which is an approximation to the expected risk, is minimizing the empirical risk to make decisions. The empirical risk based on the 0/1 loss function is defined as

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

Note that the empirical risk should asymptotically approach the expected risk.

According to Cherkassky,[[6]](#footnote-6) there are a number of inductive principles to learning:

1. Empirical Risk Minimization
2. Structural Risk Minimization
3. Penalization
4. Bayesian Inference
5. Minimum Description Length

The author primarily focuses on the four inductive principles of Penalization, Bayesian Inference, Structured Risk Minimization, and Minimum Description Length, but he gives a short description of empirical risk minimization (to be discussed later).

Cherkassky repeatedly highlights that there is an important distinction between inductive principles and learning methods. For a given inductive principle, many different learning methods (even infinitely many) can be used to execute the inductive principle. These different learning methods correspond to different classes of approximation functions (the decision rule g(x)) and the optimization technique (how the g(x) is found).

An inductive methods is concerned with a general approach to using data for learning, and the learning method that one uses from a particular inductive method is the implementation that gives the estimate.

Cherkassky segregates inductive learning methods into classical and adaptive, and he distinguishes the two inductive methods based on the amount of information available prior to learning or the amount of information assumed as valid prior to learning, Figure 1.

The classical inductive methods (e.g. empirical risk minimization) assume strong a-priori knowledge of the probability distributions whereas the adaptive inductive methods assume weak a-priori knowledge of the probability distributions. In some cases, the author believes that Bayesian is closely aligned with empirical risk minimization because Bayesian inference often assumes that prior information and the likelihood distribution of the samples is known. That is this author’s speculation, and this author needs to read considerably more information about these induction methods prior to concluding that Bayesian induction is similar to empirical induction. Cherkassky states that the penalization inductive method can be formulated in terms of Bayes induction, the author concludes otherwise[[7]](#footnote-7).



Figure 2: Inductive Principles to Learning

To illustrate the above concepts, the author Cherkassky illustrates empirical risk minimization (ERM) and the various learning methods that can be implemented under the rubic of ERM: maximum likelihood, linear regression, polynomial, and fixed-topology neural networks.

In ERM, the model is given, e.g. Gaussian density function, and then the parameters for that distribution are found, e.g. mean and standard deviation. ERM works well when the number of training samples is large relative to the model complexity (number of free parameters such as mean and standard deviation).

For example, if only one free parameter is required to be estimated, e.g. the sample mean, and if many samples are available for estimating the sample mean, then maximum likelihood is a good estimator for the mean. This good estimator is only true if the assumed model (Gaussian) that is used in the estimator is the correct density model. If however, the number of free parameters (mean, covariance matrix) that must be estimated is large relative to the number of training samples, then the ERM methods performs poorly. Moreover, the mean might be an insufficient statistic (not well estimated by the sample mean) for the model.

#### Empirical Risk Minimization

Empirical risk minimization is the inductive approach associated with the classical statistical approaches of “parametric estimation” and “classification” as developed by R.A. Fisher (1890-1962).

R.A. Fisher was trained as a mathematician, and as a student, he had not formally studied statistics. He became involved in statistics while working at the Rothamsted Experimental Station (one of the oldest agriculture research institutes in the world) Harpenden, Herfordshire, England in the year 1919, and while performing agriculture crop research, Fisher used statistics to support his studies in crop variation. His researches supported his interest in and growing passion for eugenics. During this time, he became especially involved in statistics and the design of experiments after reading William Gosset’s paper on the Student’s t- finite sampling distribution. Later, he became interested in curve fitting after reading Karl Pearson’s method of moments for curve fitting, which Fisher questioned as an appropriate method, and Fisher recast Pearson’s curve fitting approach with the method maximum likelihood.[[8]](#footnote-8) Fisher nearly single-handedly created modern empirical statistical theory, which falls under the inductive method of empirical risk minimization.

Under the rubic of empirical risk minimization, we learn that maximum likelihood is a specific “constructive method” used to estimate the parameters to an assumed probability density.[[9]](#footnote-9) For example, in maximum likelihood, a distribution is assumed for the data under question, and the parameters to the assumed distribution are estimated based on “empirically” collected data (empirical in that the data is collected from an experiment).

Maximum likelihood is a particular learning method that minimizes the Kullback-Leiber divergence metric (as used in information theory)[[10]](#footnote-10).

#### Structural Risk Minimization

Structural risk minimization was first presented in a paper by Vladimir Vapnik and Alexey Chervonenkis[[11]](#footnote-11) in 1974 as a method of optimal learning that combines “empirical risk minimization” along with an added constraint of minimizing the complexity of the model (that is used to make decisions.[[12]](#footnote-12) In essence, the theory favors less complex models.

Toward the goal of generalized learning, the author’s formulated a theory of statistical learning in which they control the empirical risk and the generalization ability of a learning machine by two factors: error-rate on training data and the capacity of the learning machine according to its Vapnik-Chervonenkis (VC) dimension[[13]](#footnote-13):

|  |  |  |
| --- | --- | --- |
|  |  | (15) |

Where the confidence interval is related to the VC dimension (h) of the decision function class and the number of training samples (n) as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

Note, for this to hold, n > h. The confidence interval holds with probability 1-.

The general intuition behind the statistical learning theory is that a researcher can design the learning machine with a high complexity decision boundary such that the empirical training errors goes to zero, which is generally not desirable. In this case, the confidence interval could be quite large, and the performance of the learning machine could be poor, in other words, the classifier error >> expected risk.

At the other extreme, the researcher could lower the confidence interval to nearly zero at the risk of increasing the empirical error which gives a learning machine poor performance, again in other words, the error >> expected risk[[14]](#footnote-14).

Vapnik and Chervnonkis recommend designing the learning machine so as to find a balance between confidence and empirical risk.

Using the statistical learning theory of Vapnik and Chervnonkis, the induction principle of structural risk minimization leads to support vector machines when the classes are separable in some feature space and the separation of the classes in the feature space are separated by a learning machine function f(x,w) such that the distance between the nearest samples between the classes is maximized[[15]](#footnote-15).

As an outcome of this theory, the learning method (the constructive method) of support vector machines was developed which will be discussed separately in a section following inductive methods.

#### Penalization

The risk[[16]](#footnote-16) used in the inductive principle of penalization is determined as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

The empirical risk denotes training error, and the penalty function is a non-negative functional associated with each possible estimate of f(x,w). The parameter weights the penalty relative to the empirical risk in decision making.

#### Minimum Description Length

This principle[[17]](#footnote-17) contrasts with all other methods in that the approach regards models as codes, as in information theory. The approach is to encode the training data such that the code length represents the data in a way that generalizes the learning machine.

#### Bayes Induction

Bayes induction principle[[18]](#footnote-18) is based on having available a-priori probability information along with a-priori defined likelihood functions. Given the a-priori probabilities, a classifier is defined that reduces empirical risk. Assuming that the loss function is the 0/1 loss, then the Bayesian induction produces a learning machine based on Bayes’ principle

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

And this minimized the expected risk. The decision rule is defined as

|  |  |  |
| --- | --- | --- |
|  |  | (19) |

## Support Vector Machines

The support vector machine (SVM) is a learning machine that belongs to the inductive class of learning known as structural risk minimization, as described above. The SVM is primarily a two-class classifier that can be extended to multiple-classes. The optimization criterion is the margin of separation between the features of the classes which ensures the minimal VC dimension and minimizes the empirical risk. Training a SVM is equivalent to solving a quadratic programming problem with linear constraints.[[19]](#footnote-19)

For classification without training error (that which minimizes the empirical error), the following linear functional defines a hyper-plane between classes that has no error (assuming the classes are separable):

|  |  |  |
| --- | --- | --- |
|  |  | (20) |

Where and is a feature vector. The classifier derived from the above functional is simply

|  |  |  |
| --- | --- | --- |
|  |  | (21) |

The learning machine is defined by both learning the vector w (the vector perpendicular to the hyperplane separating the classes) and the parameter b (the position of the hyperplane). A support vector maximizes the distance between the nearest neighbors (support vectors) of the classes with the decision boundary. The boundary of the decision surface, the hyperplane, is given as

|  |  |  |
| --- | --- | --- |
|  |  | (22) |



Figure 3: Hyperplane wx + b (defined by vector w) and margin

The distance between the classes is given by two examples and that lie on the boundaries and . The distance is determine as

|  |  |  |
| --- | --- | --- |
|  |  | (23) |

The maximization of the distance across the margin, reducing VP dimension, is done by quadratic programming for

|  |  |  |
| --- | --- | --- |
|  |  | (24) |

We are unable to solve this directly, so to solve the minimization, we use Langrangian multipliers to optimize to find w and b as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (25) |

Prior to solving this optimization directly, we will first map the feature vector x non-linearly via a function into a higher (possibly much higher) dimensional feature space in which the classes become linearly separable. The function is defined as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (26) |

One now develops the learning algorithm in the space rather than original space with samples defines as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (27) |

Where . We now substitute for x into the Langrangian

|  |  |  |
| --- | --- | --- |
|  |  | (28) |

To optimize, we maximize with respect to w and b, and then we minimize with respect to the Langrangian multipliers . [[20]](#footnote-20) Taking the partials of the Langrangian and setting to zero to find the extremes, we have

|  |  |  |
| --- | --- | --- |
|  |  | (29) |

And

|  |  |  |
| --- | --- | --- |
|  |  | (30) |

Solving the first partial gives us

|  |  |  |
| --- | --- | --- |
|  |  | (31) |

Which reduces to

|  |  |  |
| --- | --- | --- |
|  |  | (32) |

Solving the second partial gives us

|  |  |  |
| --- | --- | --- |
|  |  | (33) |

Which reduces to

|  |  |  |
| --- | --- | --- |
|  |  | (34) |

This is an amazing result. Substituting w into the Langrangian and maximizing based on the Langrangian multipliers we get the following quadratic optimization problem

|  |  |  |
| --- | --- | --- |
|  |  | (35) |

Substituting a kernel function for the non-linear mapping functions , we get the quadratic optimization problem:

|  |  |  |
| --- | --- | --- |
|  |  | (36) |

With the following linear constraints

|  |  |  |
| --- | --- | --- |
|  |  | (37) |

And

|  |  |  |
| --- | --- | --- |
|  |  | (38) |

In matrix notation, the above simplifies to

|  |  |  |
| --- | --- | --- |
|  |  | (36) |

Under the constraints

|  |  |  |
| --- | --- | --- |
|  |  | (36) |

Where

## Kernel Functions

# Project

This project consists of several experiments to compare various versions of the eigenface recognition algorithm on different data sets.

## Experiment 1: SVM

### 16x20 Images

#### Polynomial Kernels

#### RBF Kernels

### 48x60 Images

#### Polynomial Kernels

#### RBF Kernels

## Experiment 2: Bayesian

### Training Parameters

### Testing on 16x20 Images

### Testing on 48x60 Images

# Conclusion

## Part A: SVM

## Part B: Bayesian

## Comparison

# Contributors

1. Cherkassky, Vladimir, “Inductive Principles for Learning from Data,” http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.26.7820&rep=rep1&type=pdf [↑](#footnote-ref-1)
2. M. Turk and A. Pentland, “Eigenfaces for Recognition,” Journal of Cognitive Neuroscience, vol. 3, no. 1 pp. 71-86, 1991. [↑](#footnote-ref-2)
3. L Sirovich and M Kirby, “Low-dimensional procedure for the characterization of human faces,” Journal of the Optical Society of America A, 4(3), 519-524. [↑](#footnote-ref-3)
4. Cherkassky, Vladimir, “Inductive Principles for Learning from Data,”page 2. [↑](#footnote-ref-4)
5. Klaus-Robert Muller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernard Scholkoph, “An Introduction to Kernel-Based Learning Algorithms,” IEEE Transactions on Neural Networkds, Vol. 12, No. 2, March 2001. [↑](#footnote-ref-5)
6. Ibid, page 5. (Cherkassky, Vladimir, “Inductive Principles for Learning from Data”) [↑](#footnote-ref-6)
7. Ibid, page 13. (Cherkassky, Vladimir, “Inductive Principles for Learning from Data”) [↑](#footnote-ref-7)
8. Spanos, Arias, “R. A. Fisher: how an outsider revolutionized statistics,” http://errorstatistics.com/2014/02/17/r-a-fisher-how-an-outsider-revolutionized-statistics-2/ [↑](#footnote-ref-8)
9. Ibid, page 8. (Cherkassky, Vladimir, “Inductive Principles for Learning from Data” ) [↑](#footnote-ref-9)
10. Kevin Swersky, “Inductive Principles for Learning Restricted Boltzmann Machines,” Master’s Thesis, University of British Columbian, August 2010. [↑](#footnote-ref-10)
11. http://en.wikipedia.org/wiki/Structural\_risk\_minimization [↑](#footnote-ref-11)
12. Bebis, George, Lecture on “Support Vector Machines,” page 7, 2015. http://www.cse.unr.edu/~bebis/CS479/ [↑](#footnote-ref-12)
13. Corinna Cortes and Vladimir Vapnik, “Support-Vector Networks,” Machine Learning, 20, pages 273-297, 1995. [↑](#footnote-ref-13)
14. Klaus-Robert Muller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernard Scholkoph, “An Introduction to Kernel-Based Learning Algorithms,” IEEE Transactions on Neural Networkds, Vol. 12, No. 2, March 2001. [↑](#footnote-ref-14)
15. Vladimir Vapnik, “The Nature of Statistical Leaning Theory,” Springer-Verlag, New York, 1995. [↑](#footnote-ref-15)
16. Cherkassky, Vladimir, “Inductive Principles for Learning from Data,”page 8. [↑](#footnote-ref-16)
17. Ibid, page 13. (Cherkassky, Vladimir, “Inductive Principles for Learning from Data.”) [↑](#footnote-ref-17)
18. Ibid, page 11. (Cherkassky, Vladimir, “Inductive Principles for Learning from Data.”) [↑](#footnote-ref-18)
19. Bebis, George, Lecture on “Support Vector Machines,” page 7, 2015. http://www.cse.unr.edu/~bebis/CS479/ [↑](#footnote-ref-19)
20. Klaus-Robert Muller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernard Scholkoph, “An Introduction to Kernel-Based Learning Algorithms,” IEEE Transactions on Neural Networkds, Vol. 12, No. 2, March 2001. [↑](#footnote-ref-20)