Engineering and Applied Science Programs for Professionals
Whiting School of Engineering
Johns Hopkins University
685.621 Algorithms for Data Science
Machine Learning III

This document provides a rollup of the Machine Learning II. In this module the classifiers are two-class classifiers, including Fisher linear discrimination, expectation maximization (EM) with Bayes decision theory, nonparametric classifiers, k-nearest neighbors and Parzen windows. Additionally, a nonlinear kernel based methods are shown, kernel Fisher's discriminant (KFD), Probabilistic Neural Networks (PNN) and Support Vector Machine (SVM).

Contents

1 Classification

Machine learning for a classification task involves training over a set of samples $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_\lambda]$ where $\mathbf{x} \in \Upsilon_n$. Where the symbol Υ is used to represent the n-dimensional space the features in x resides. Each sample in the training set contains one target value $\mathbf{C} = C_j = [C_1, C_2, \cdots, C_c], j = 1, 2, \cdots, c$, (known as the class labels $y_i \in \mathbf{C}, i = 1, 2, \cdots, m$) which describes the class to which the sample is a member of. The objective is to separate the data into their classes such that the degree of association is strong between the data sets of the same class and weak between members of different classes. From the class separation, an unseen sample $\mathbf{x}_0 \in \Upsilon_n$ can then be appropriately classified. In this section 5 classification methods are presented, Fisher's linear discriminant, expectation maximization with mixture models (EM), k-nearest neighbors (k-NN), kernel Fisher's discriminant (KFD), and Parzen window.

1.1 Fisher's Linear Discriminant (FLD)

The Fisher's Linear Discriminant (FLD) is used to quantify two-class separability of features (Fisher, 1936). FLD is a simple technique which measures the discrimination of sets of real numbers. Without going into all of the theory of the FLD lets focus on the primary components assuming we have a two class problem, equal class sample and a covariance matrix that is generated from normal distributions. The within-class scatter matrix is defined as

$$S_W = \sum_C P_C S_C \tag{1}$$

where S_C is the covariance matrix for class $\mathbf{C} \in \{-1, +1\}$

$$S_C = \sum_{\substack{i=1,\\ i \in C}}^{l_C} (\mathbf{x}_i - \mu_C)(\mathbf{x}_i - \mu_C)^T \tag{2}$$

and P_C is the *a priori* probability class $\mathbf{C}$. That is, $P_C \approx k_C/k$, where k_C is the number of samples in class $\mathbf{C}$, out of a total of k samples. The between-class smatter matrix is defined as

$$S_B = \sum_{\mathbf{C}} (\mu_{-1} - \mu_{+1})(\mu_{-1} - \mu_{+1})^T \tag{3}$$

where μ is the global mean vector

$$\mu = \frac{1}{l} \sum_{i=1}^{l} \mathbf{x}_i \tag{4}$$

and the class mean vector μ_C is defined as

$$\mu_C = \frac{1}{l_C} \sum_{\substack{i=1,\\ i \in C}}^{l_C} \mathbf{x}_i \tag{5}$$

Now lets look at the criterion function $J(\cdot)$ written as follows:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \tag{6}$$

where $\mathbf{w}$ is calculated to optimize $J(\cdot)$ as follows:

$$\mathbf{w} = S_W^{-1}(\mu_{-1} - \mu_{+1}) \tag{7}$$

w for the Fisher Linear Discriminant has been obtained, which will allow for the linear function to yield the maximum ratio between of the between-class scatter and the within-class scatter. It should be noted that in (Bishop, 2006) the two classes are set as 1 and 2. Now let's determine a threshold b that will allow us to determine which

class a new observation will belong to. The optima decision boundary assuming each class has the same number of samples can be calculated as follows:

$$b = -0.5(\mathbf{w}\mu_{-1} + \mathbf{w}\mu_{+1}) \tag{8}$$

Now, if we have a new input observation x we can determine which class the new observation belongs to based on the following

$$y = \mathbf{w}\mathbf{x} + b \tag{9}$$

where $y < 0$ is class -1 and $y \geq 0$ is class $+1$.

The previous discussion is based on the FLD and is simplified as a two class linear discriminant function presented in (Bishop, 2006) Section 4.1.1 Two classes. Credit is given to Fisher for his work in this area of linear discrimination.

1.2 Expectation Maximization

The idea behind the EM algorithm (Dempster et al., 1977) is that even though the data values of $\mathbf{x}$, feature vectors $\mathbf{x}_n \in \Upsilon_n$, are unknown/incomplete the distribution $f(\mathbf{x}|p)$ can be used to determine an estimate for the maximum likelihood (Tomasi, 2006). In maximum likelihood estimation, the estimate to be modeled is the parameter(s) for which the observed data are the most likely. This is done by iteratively estimating the data parameters, then using the data to update the estimated parameters, until a desired convergence is met. The two major steps of the EM algorithm are the expectation step (E-Step) and the maximization step (M-Step).

The EM algorithm consists of choosing initial parameters for the means, $\mu_k^{(j)}$, standard deviations, , and mixing probabilities, $\sigma_k^{(j)}$, for a user defined number of clusters, k, then performing the E-Step and M-Step successively until convergence, where i is the current iteration and n is the number of samples. The convergence criteria is determined by examining when the parameters quit changing, i.e., when $\left|\mu_k^{(j)} - \mu_k^{(j+1)}\right| < \epsilon$ & $\left|\sigma_k^{(j)} - \sigma_k^{(j+1)}\right| < \epsilon$ & $\left|p^{(j)}(k|l) - p^{(j+1)}(k|l)\right| < \epsilon$ for some epsilon (ϵ) and distance calculation (Euclidian distance). The maximum likelihood estimation is a method of estimating the parameters of the distributions based upon the observed data.

The expectation step (E-Step) calculates the membership probabilities, $p(k|l)$ (Tomasi, 2006). The mixing probabilities p_k are viewed as the sample mean of the membership probabilities $p(k|l)$ assuming a uniform distribution over all the data points. The Gaussian function, $g(\mathbf{x}; \mu_k^{(j)}, \sigma_k^{(j)})$, is used to compute mixture of Gaussian functions as shown in the denominator of $p(k|l)$.

$$p^{(j)}(k|l) = \frac{p_k^{(j)} g(\mathbf{x}; \mu_k^{(j)}, \sigma_k^{(j)})}{\sum_{k=1}^{K} p_k^{(j)} g(\mathbf{x}; \mu_k^{(j)}, \sigma_k^{(j)})} \tag{10}$$

$$g(\mathbf{x}; \mu_k^{(j)}, \sigma_k^{(j)}) = \frac{1}{\left(\sqrt{2\pi}\sigma_k\right)^n} \exp\left\{ -\frac{1}{2}\left(\frac{\|\mathbf{x} - \mu_k\|}{\sigma_k}\right)^2 \right\} \tag{11}$$

The maximization step (M-Step) uses the data from the expectation step as if it were measured data to determine the maximum likelihood estimate of the parameter (Tomasi, 2006). This estimated data is often referred to as the "imputed" data. This step is dependent upon the membership probabilities $p(k|l)$ which are computed in the E-Step. The EM algorithm consists of iterating the mean, standard deviation, and mixing probabilities until convergence. The mixing probabilities are the sample mean of the conditional probabilities $p(k|l)$ assuming a uniform distribution over all the data points.

$$\mu_k^{(j+1)} = \frac{\sum_{i=1}^{l} p^j(k|i)\mathbf{x}_i}{\sum_{i=1}^{l} p^j(k|i)} \tag{12}$$

$$\sigma_k^{(j+1)} = \sqrt{\frac{1}{D}\frac{\sum_{i=1}^{l} p^j(k|i)\left\|\mathbf{x}_i - \mu_k^{(j+1)}\right\|^2}{\sum_{i=1}^{l} p^j(k|l)}} \tag{13}$$

$$p_k^{(j+1)} = \frac{1}{l}\sum_{i=1}^{l} p^j(k|i) \tag{14}$$

1.2.1 Mixture Models

In mixture models, also known as model-based Gaussian clustering, the multivariate Gaussian normal is used as a density function similarly described in Equation 11. The general multivariate normal density for n dimensions is

$$g(\mathbf{x}; \mu_k, \Sigma_k) = \frac{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k)\right\}}{\left(\sqrt{2\pi}\right)^n\left|\Sigma_k\right|^{1/2}}. \tag{15}$$

The geometric characteristics (size, shape and orientation) of the clusters are determined by the covariance matrix Σ_k which is generated in terms of eigenvalue decomposition described in Martinez and Martinez (2002). The decomposition of the covariance matrix Σ_k is used as a suitable model for the geometric characteristics of the cluster. The structure of the covariance matrix is as follows:

$$\Sigma_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T \tag{16}$$

where λ_k is a scalar, D_k is the orthogonal matrix of eigenvectors and A_k is a diagonal matrix whose elements are proportional to the eigenvalues of Σ_k. Note that in EM the values p_k, μ_k, and σ_k are updated after each iteration and in the mixture models σ_k is replaced by Σ_k to represent the geometric characteristics of the clusters.

The eigenvalue decomposition can be modeled as various clustering arrangements. Celeux and Govaert (1995), describe in detail fourteen models based on the eigenvalue decomposition. Allowing for variations in the orientation, volume, shape and size of the clusters; six of these models are shown in Table 1 (Martinez and Martinez, 2002).

Table 1: Parameterization for Mixture Models

Model	Σ_k	**Geometric Shapel**	**Volume**	**Shape**	**Orientation**
1	$\lambda\mathbf{I}$	Spherical	Equal	Equal	N/A
2	$\lambda_k\mathbf{I}$	Spherical	Variable	Equal	N/A
3	$\lambda\mathbf{D}\mathbf{A}\mathbf{D}^T$	Ellipsoid	Equal	Equal	Equal
4	$\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T$	Ellipsoid	Variable	Variable	Variable
5	$\lambda\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T$	Ellipsoid	Equal	Equal	Variable
6	$\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T$	Ellipsoid	Variable	Equal	Variable

The eigenvalue decomposition can be modeled as various clustering arrangements, i.e., spheres, ellipsoids and rotations of ellipsoids. Allowing the orientation, volume, shape and size of the clusters define the various models used. Figure 1 shows the mixture model using rotated ellipsoids (Model 4) to generate the decision boundary around each class.

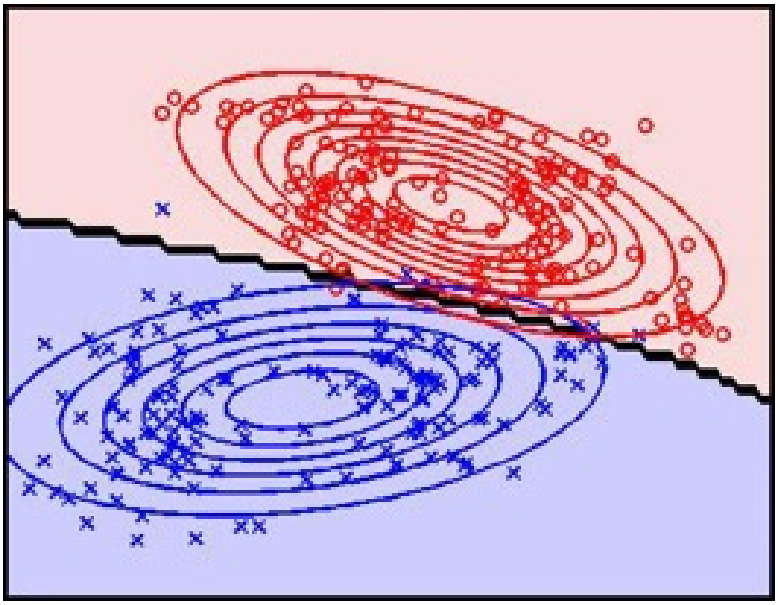

Figure 1: Expectation Maximization using mixture models with Decision Boundary

1.2.2 Bayes Classifier

The EM algorithm can be used to find a class label for an input sample. Classification uses input samples described by feature vectors $\mathbf{x}_0 \in \Upsilon_n$ to assign the samples to a given class $\mathbf{C} = C_j = [C_1, C_2, \cdots, C_c], j = 1, 2, \cdots, c$. The Bayes classifier extends a general multivariate normal case where the covariance matrix Σ_j for each class is different. For the multi-class classifier each class must have individual conditional probability densities where the densities are modeled as normal distributions. The classes C_j are defined as normal distributions centered about the mean vector μ_j. The mean vector, μ_j, and the covariance matrix, Σ_j, are calculated using the EM algorithm. The vector $\mathbf{x}_0$ is a n-dimensional vector of the observed data, and $|\Sigma_i|$ and Σ_i^{-1} are the determinants and inverse covariance matrix of the given class. The posterior probability of class membership can be calculated by Bayes rule if C_j is defined as the event of belonging to population j. Using the density function $g(\mathbf{x}; \mu_k^{(i)}, \sigma_k^{(i)})$ (Tomasi, 2006), the Bayes classifier can be expressed in terms of the prior probabilities, $P(C_i)$, and posterior probability of class membership as follows:

$$P(C_j|x_0) = \frac{P(C_j)\frac{1}{\sqrt{(2\pi)^n|\Sigma_j|}}\exp\Big[-\frac{1}{2}\big(\mathbf{x}_0-\mu_j\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0-\mu_j\big)\Big]}{\sum_{i=1}^{c}P(C_i)\frac{1}{\sqrt{(2\pi)^n|\Sigma_i|}}\exp\Big[-\frac{1}{2}\big(\mathbf{x}_0-\mu_i\big)^T\Sigma_i^{-1}\big(\mathbf{x}_0-\mu_i\big)\Big]} \tag{17}$$

where the a priori probabilities P(Cj) are the estimates of belonging to a class and under the assumption that $\Sigma_j = \Sigma$ for $\forall j$.

1.3 k-Nearest Neighbors

k-Nearest Neighbors, Figure 2, is a lazy learning approach that compares new samples with all of the samples in the training set, looking for the kth nearest (Cover and Hart, 1967; Duda et al., 2001; Bishop, 2006).

Let the vectors $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_\lambda] \in \Upsilon_n$ and class labels $y_i \in \mathbf{C} = [C_1, C_2, \cdots, C_c], c \in \Re, i = 1, 2, \cdots, \lambda$, be a set of training vectors. Given an unknown feature vector $\mathbf{x}_0$ and a distance measure, the algorithm for the k-nearest neighbor rule is as follows (Theodoridis and Koutroumbas, 2006):

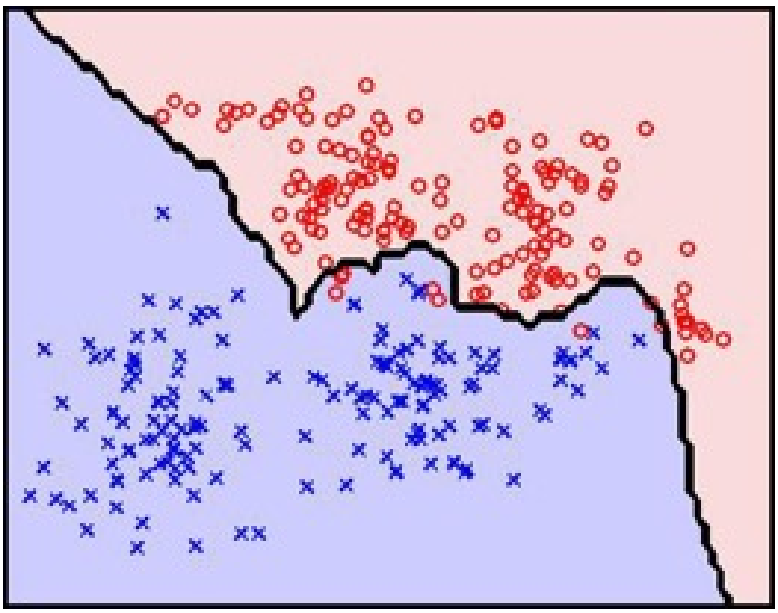

Figure 2: k-NN Decision Boundary

- Out of the λ training vectors $\mathbf{x}$, identify the k-nearest neighbors, irrespective of class label. k is chosen to be odd for a two-class problem, and in general not to be a multiple of the number of classes.
- Out of the k samples, identify the number of vectors, k_j, that belong to class $\mathbf{C}$, where $\sum_j k_j = k$.
- Assign $\mathbf{x}_0$ to the class $\mathbf{C}$ with the maximum number of k_j of samples.

The distance measures used from the feature $\mathbf{x}_i$ to each of its k-nearest neighbors include the Euclidian and Mahalanobis. The advantage of k-nearest neighbor is the simplicity of the assignment procedure. The disadvantage of the method lies in the necessity to store all samples and compare each with an unknown sample (Fukunaga, 1990).

1.4 Kernel Fisher's Discriminant

The kernel Fisher discriminant is the non-linear extension of the linear FLD (Jaakola and Haussler, 1998; Mika et al., 1999; Scholkopf and Smola, 2002). In the linear case, Fisher's discriminant is computed by maximizing the coefficients of the following equation

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \tag{18}$$

To use the Fisher's discriminant for nonlinearly separable data (Mika, et al. 1999) map the input feature space with the use of a kernel. The input space is represented by a training set $\mathbf{x}_i$ of vectors with a feature dimensionality of n. The corresponding class labels are represented as $y_i \in \mathbf{C}$, where $\mathbf{C} = [C_{-1}, C_{+1}] = [-1, +1], i = 1, 2, \cdots, \lambda$ and λ is the training set size. The basic idea is to first map the input features from the input space to the kernel space via a kernel function and then perform linear FLD. The aim is to find a direction $\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i)$ from the feature space to the kernel space given by alpha vectors $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_\lambda]^T$ (Mika et al., 1999). Using the definitions of SB and SW the Fisher's linear discriminant in the mapped feature space can be defined as

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T M \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T N \boldsymbol{\alpha}} \tag{19}$$

where $M = (M_{-1} - M_{+1})(M_{-1} - M_{+1})^T$ is a $[\lambda]$ matrix,

$$M_{-1} = \frac{1}{l_{-1}} \sum_{\substack{j=1, \\ j \in C_{-1}}}^{l_{-1}} K(\mathbf{x}_i, \mathbf{x}_j) \tag{20}$$

$$M_{+1} = \frac{1}{l_{+1}} \sum_{\substack{j=1, \\ j \in C_{+1}}}^{l_{+1}} K(\mathbf{x}_i, \mathbf{x}_j) \tag{21}$$

and

$$N = \sum_{\substack{j=1, \\ j \in \mathbf{C}}}^{l_{\mathbf{C}}} K(\mathbf{x}_i, \mathbf{x}_j)\Big(I - \frac{1}{l_{\mathbf{C}}}\Big)K(\mathbf{x}_i, \mathbf{x}_j)^T \tag{22}$$

where $\mathbf{C} = \{C_{-1} - C_{+1}\} = \{-1, +1\}$.

In (Mika et al., 1999), numerical issues and regularization are discussed regarding the calculation of Equation 22. This is resolved by simply adding a multiple of the identity matrix to N defined as:

$$N_\mu = N + \mu I. \tag{23}$$

The next step is to use the alpha vectors and the kernel matrix to project the $n-1$ dimensional input feature space into a one dimensional space as follows:

$$\hat{\mathbf{x}} = K(\mathbf{x}_i, \mathbf{x}_j)\alpha. \tag{24}$$

The projection in Equation 23 now becomes the space that is to be solved using an optimization solution to maximize the margin of separation between classes as shown in Figure 3. In (Mika et al., 1999), the Matlab Optimization Toolbox (2004) is used to solve the optimization problem (Scholkopf and Smola, 2002) with the projected space calculated in Equation 24. In this research the one dimensional SMO (Franc and Hlavac, 2007) is used as the optimization solution. This results in the non-negative alpha vectors $\hat{\alpha}_i = (\hat{\alpha}_1, \hat{\alpha}_2, \cdots, \hat{\alpha}_l)$ with an upper bound $\hat{C}, \hat{C} \geq \hat{\alpha}_l \geq 0$. The support vectors for the KFD trained model are $\mathbf{x}_k = \mathbf{x}_i$ and the decision function of the KFD classifier is written as $\text{sign}(f(\mathbf{x}))$ where $f(\mathbf{x})$ is defined by:

$$f(\mathbf{x}) = \mathbf{w}\phi(\mathbf{x}) + b = \sum_{i=1}^{l} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b. \tag{25}$$

This is equivalent to the maximal margin hyperplane in the input space defined by the kernel (Cristianini and Shawe-Taylor, 2000). The goal of the KFD is to solve for $\boldsymbol{\alpha}$ and the bias b. To compute the bias b, Equation 33 is rewritten as follows:

$$\sum_{i=1}^{l_S} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b = y_i. \tag{26}$$

Therefore, the bias is calculated by obtaining the average as (Scholkopf and Smola, 2002):

$$b = \frac{1}{l} \sum_{i=1}^{l} \left(y_i - \sum_{i=1}^{l_S} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b \right) \tag{27}$$

In order to reduce the number of false positives and false negatives the optimal bias in Equation 35 can be adjusted accordingly. In this case the bias is a threshold (Scholkopf and Smola, 2002).

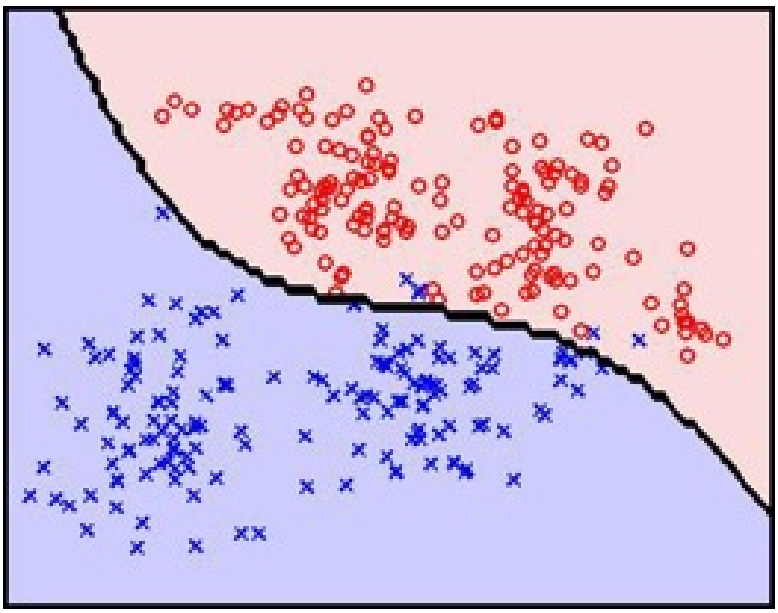

Figure 3: KFD Decision Boundary using RBF Kernel

1.5 Parzen Window

Parzen estimation is a refinement of histogramming (Parzen, 1962; Fukunaga, 1990; Duda et al., 2001; Bishop, 2006). The basic idea behind Parzen window estimation is that the knowledge gained by each training sample $\mathbf{x}$ of the input space, X_n, is represented by a function centered at $\mathbf{x}$ in the feature space. The functions themselves are represented with the use of a distance measure or a kernel estimator. The final class estimation is derived by summing the results from the kernel functions of each training sample:

$$p_k = \frac{1}{l_k} \sum_{i=1}^{l_k} K(\mathbf{x}, \mathbf{x}_i). \tag{28}$$

For example, the Parzen window density model is optimized by maximizing the likelihood of the training data with the use of a Gaussian window surrounding each input data point. The Gaussian window can be represented with the use of a kernel function $K(\mathbf{x}, \mathbf{x}_i)$ as an interpolation function which defines an inner product between the individual training sample. The Radial Basis kernel function uses a window width parameter,σ, which is also known as the spread of the function:

$$p_k = \frac{1}{l_k} \sum_{i=1}^{l_k} \frac{1}{\sqrt{(2\pi)^{l_k}} \sigma^{l_k}} \exp\left(- \frac{\|\mathbf{x} - x_i\|^2}{2\sigma^2} \right). \tag{29}$$

This results in a sum of small multivariate Gaussian probability distributions centered at each training sample $\mathbf{x}$, an example is shown in Figure 4. As the density of the training samples and their respective Gaussian distributions increase the estimation of the probabilities approach the true probability density function (PDF) of the training samples. The estimation for classification for a data cluster is then based on a threshold set for the combined posterior probability from all samples. The classification decision assigns the samples to the class with maximal posterior probability according to the inequality:

$$\frac{1}{l_k} \sum_{i=1}^{l_k} \frac{1}{\sqrt{(2\pi)^{l_k}} \sigma^{l_k}} \exp\left(- \frac{\|\mathbf{x} - x_i\|^2}{2\sigma^2} \right) > \frac{1}{l_j} \sum_{i=1}^{l_j} \frac{1}{\sqrt{(2\pi)^{l_j}} \sigma^{l_j}} \exp\left(- \frac{\|\mathbf{x} - x_i\|^2}{2\sigma^2} \right), \forall k \neq j \tag{30}$$

This method requires a reasonably large training data set and is computationally inexpensive during training but is computationally expensive for testing. During testing the kernel function must be computed for each of the training samples making a comparison between the new sample $\mathbf{x}_0$ and all of the existing training samples $\mathbf{x}$. Several kernel approaches have been proposed in literature (Fukunaga, 1990; Wand and Jones, 1995). The kernels were originally presented by Parzen (1962).

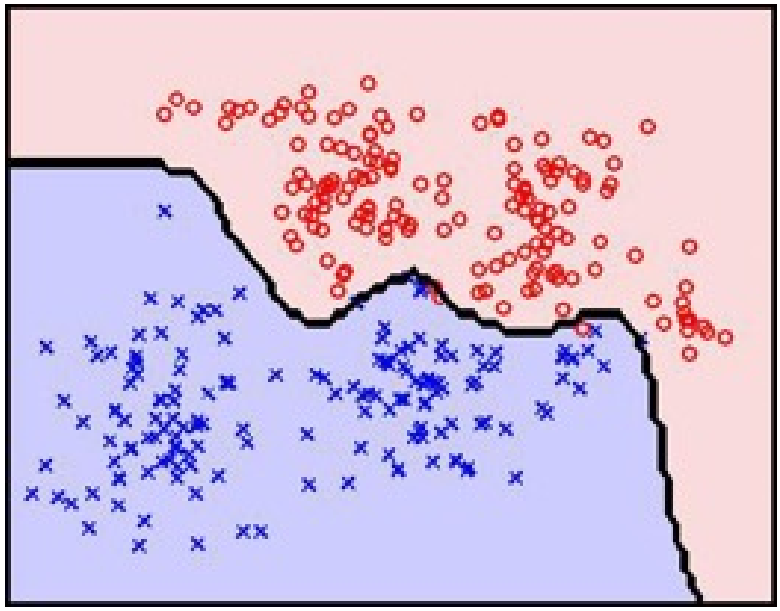

Figure 4: Parzen Density Estimator with RBF window with Decision Boundary

1.6 Probabilistic Neural Networks (PNN)

The classification frame work of the probabilistic neural network is shown in Figure 5 (Specht, 1998; 1990). There are a few decisions that have to be made regarding training of the neural network. First, the number of training samples and number of classes are selected for the pattern layer; this defines the structure of the network. For example, the set of input training samples is represented as $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\lambda]^T \in \Upsilon^n$ and a class label $y_i \in C = [C_1, C_2, \ldots, C_c], i = 1, 2, \ldots, \lambda$. This will result in c groups with each group in the pattern layer containing λ neurons. Second, for the summation layer the smoothing parameter, σ, in the nonlinear operation $f(\mathbf{z}_i)$ of the neural network must be determined. As a general guideline the value of the smoothing parameter, σ, should be chosen as a function of the dimension of the problem, n, and the number of training samples, λ (Specht, 1990). The structure of the probabilistic neural network classifier has three layers as shown in Figure 5, pattern layer, summation layer and the decision layer. The pattern layer forms a dot product of the input features, $\mathbf{x}$, with the weight vectors, w_i, resulting in $z_i = \mathbf{x} \bullet \mathbf{w}_i$. A nonlinear operation $f(\mathbf{z}_i)$ on $\mathbf{z}_i$ is preformed prior to outputting the activation to the summation level.

$$f(\mathbf{z}_i) = \exp\left(\frac{\mathbf{z}_i - 1}{\sigma^2}\right) \tag{31}$$

The summation layer sums the inputs from the pattern layer that corresponds to the class from which the training patterns were selected. The output layer returns the summation values for each of the c classes, a two-class example is shown in Figure 6. Each output values $P_1, \ldots, P_c$ is the posterior probability that the sample belongs to that particular class, where you $\sum_{j=1}^{c} P_j = 1$.

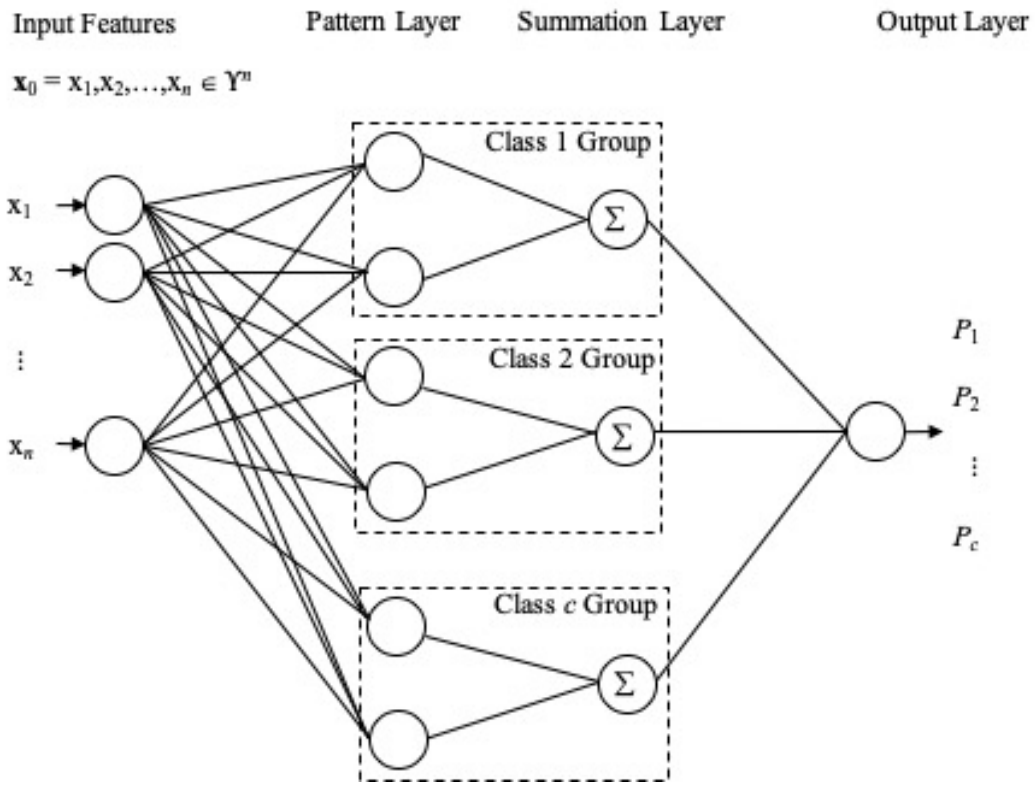

Figure 5: Probabilistic Neural Network Classification Structure

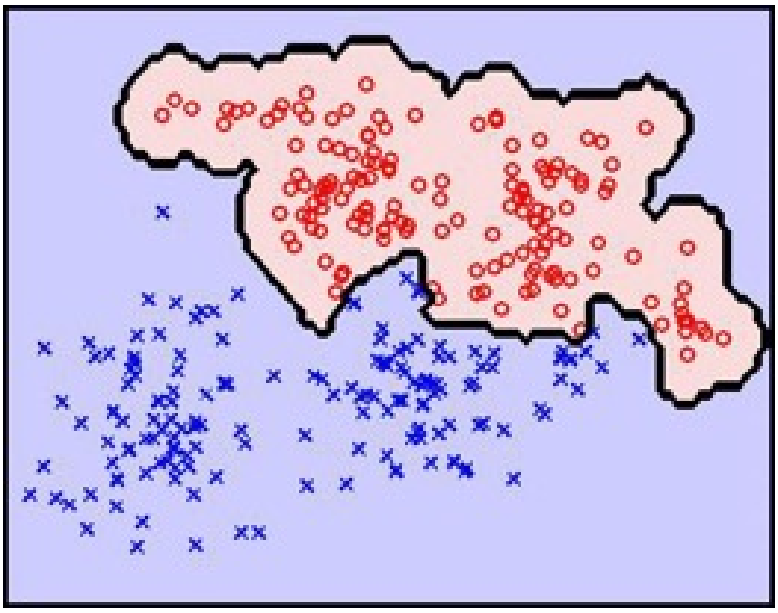

Figure 6: Probabilistic Neural Network Decision Boundary

1.7 Support Vector Machine (SVM)

SVM performs pattern recognition for two-class problems by determining the separating hyperplane that maximizes the distance between the closest points of each class in the training set (Scholkopf et al., 1998; 1999; 2002; Burgers, 1998; Vapnik, 1998; Platt, 2000; Hsu et al., 2006). These closest points are called support vectors. In finding the hyperplane, the SVM performs a nonlinear separation in the input space by using a nonlinear transformation $\phi(\mathbf{x}_i)$ that maps the data points $\mathbf{x}_i$ of the input space, Υ^n, into a potentially higher dimensional space, called kernel space $\Upsilon^\lambda (\lambda > n)$. The mapping $\phi(\mathbf{x}_i)$ is represented in the SVM classifier by a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ that defines an inner product in Υ^λ.

The optimal hyperplane is the one with the maximal distance (in space Υ^p) to the closest points $\phi(\mathbf{x}_i)$ of the training data, an example is shown in Figure 7. Determining the hyperplane requires maximizing the following function with respect to α

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{32}$$

under the constraints $\sum_j \alpha_j y_j = 1, \ldots, \lambda$. The non-negative Lagrangian multipliers are $\alpha_k = \left(\alpha_1, \ldots, \alpha_{l_s}\right)$ with an upper bound $\hat{C}$, $\hat{C} \leq \alpha_{l_s} \leq 0$. The Lagrangian multipliers are also known as the alpha vectors.

With the given support vectors $\mathbf{x}_k$ and class labels y_k, the decision function of the SVM classifier can be written as $\text{sign}(f(\mathbf{x}))$ where $f(\mathbf{x})$ is defined by:

$$f(\mathbf{x}) = \mathbf{w}\phi(\mathbf{x}) + b = \sum_{k=1}^{l_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}) + b \tag{33}$$

This is equivalent to the maximal margin hyperplane in the input space defined by the kernel (Cristianini and Shawe-Taylor, 2000). The goal of the SVM is to solve for, α, the bias b and the support vectors $\mathbf{x}_k$. To compute the bias b, Equation 33 is rewritten as follows:

$$\sum_{k=1}^{l_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_i) + b = y_i \tag{34}$$

Therefore, the bias is calculated by obtaining the average as (Scholkopf and Smola, 2002):

$$b = \frac{1}{l} \sum_{i=1}^{l} \left(y_i - \sum_{k=1}^{l_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_i) \right) \tag{35}$$

In order to reduce the number of false positives and false negatives the optimal bias in (35) can be adjusted accordingly. In this case the bias is a threshold (Scholkopf and Smola, 2002).

Solving Equation 32 is a dual quadratic programming (QP) problem. There are several methods used to solve the quadratic programming problem, including Kernel-Adatron (Friess et al., 1998; Cristianini and Shawe-Taylor, 2004), LOQO (Vanderbei and Shanno, 1999) and sequential minimal optimization (SMO) (Cristianini and Shawe-Taylor, 2000; Franc and Hlavac; Mak, 2000; Platt, 2000).

Several solutions are available as complete SVM systems to include LIBSVM (Chang and Lin, 2001), Matlab Optimization Toolbox, Weka and SVMlight (Joachims, 1998). Each of these methods has individual advantages and disadvantages that are beyond the scope of this research.

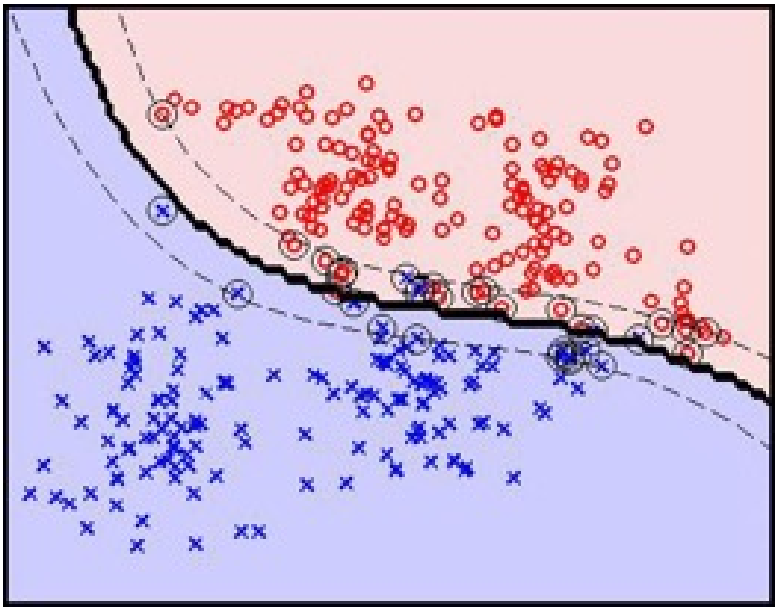

Figure 7: SVM with Optimal Hyperplane

References

[1] Bishop, Christopher M., *Neural Networks for pattern Recognition*, Oxford University Press, 1995

[2] Bishop, Christopher M., *Pattern Recognition and Machine Learning*, Springer, 2006

[3] Burgers, C., *A tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery*, Volume 2, Number 2, pp, 121-167, 1998

[4] Celeux, G. and G. Govaert, *Gaussian Parsimonious Clustering Models*, Pattern Recognition, Volume 28, pp. 781-793, 1995

[5] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronal L., and Stein, Clifford, *Introduction to Algorithms*, 3rd Edition, MIT Press, 2009

[6] Cover, T. and Hart, P. E. (1967). Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory. Volume 13, pp. 21-27, 1967

[7] Cristianini, N., and Shawe-Taylor J., *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge CB2 2RU, UK: Cambridge University Press, 2000

[8] Cristianini, N., and Shawe-Taylor J., *Kernel Methods for Pattern Analysis*, Cambridge CB2 2RU, UK: Cambridge University Press, 2004

[9] Dempster, A. P., Laird, N. M. and Rubin, D. B., *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society B, Volume 39, Number 1, pp.1–22, 1977

[10] Duda, R.O., Hart, P.E. and Stork, D.G., *Pattern Classification* (2nd. Ed.), New York, NY: John Wiley & Sons, 2001

[11] Duin, Robert P.W., Tax, David and Pekalska, Elzbieta, *PRTools*, http://prtools.tudelft.nl/

[12] Franc, Vojtech and Hlavac, Vaclav, *Statistical Pattern Recognition Toolbox*, https://cmp.felk.cvut.cz/cmp/software/stprtool/index.html

[13] Friess, T-T., Cristianini, N. and Campbell, C., *The Kernel-Adatron algorithm: a fast and simple learning procedure for support vector machines*, Proceedings On Machine Learning 15th International Conference, pp. 188-196, 1998

[14] Fukunaga, Keinosuke, *Introduction to Statistical Pattern Recognition*, Academic Press, 1972

[15] Hsu, C.-W., Chang, C.-C., and Lin, C.-J., *A practical guide to support vector classification*, Department of Computer Science and Information Engineering National Taiwan, 2006, Retrieved March 15, 2006 http://www.csie.ntu.edu.t

[16] Jaakola, T. S. and Haussler, D., *Exploring Generative Models in Discriminative Classifiers*, Advances in Neural Information Processing Systems, Kearns, M.S., Soll, S. A. and Cohn, D. A. (Eds.), Volume 11, Cambridge, MA: MIT Press, 1998

[17] Machine Learning at Waikato University, *WEKA*, https://www.cs.waikato.ac.nz/ ml/index.html

[18] Mak, G., *The Implementation of Support Vector Machines Using the Sequential Minimal Optimization Algorithm*, University published master's thesis, 2000, Retrieved October 28, 2007, http://moncs.cs.mcgill.ca/MSDL/10_people.h

[19] Martinez W. L. and Martinez, A. R., *Computational Statistics Handbook with MATLAB*, Boca Raton, FL: Chapman & Hall/CRC, 2002

[20] Mika, S., G. Ratsch, G., Weston, J., Scholkpf, B. and Muller, K., *Fisher Discriminant Analysis with Kernels*, Neural Networks for Signal Processing IX, Proceedings of the 1999 IEEE Signal Processing Society Workshop, pp. 41-48, 1999

[21] Parzen, E., *On the Estimation of a Probability Density Function and Mode*, Annals of Mathematical Statistics, Volume 33 pp. 1065-1076, 1962

[22] Press, William H., Teukolsky, Saul A., Vetterling, William T., and Flannery, Brian P., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Jan 31, 1986

[23] Press, William H., Teukolsky, Saul A., Vetterling, William T., and Flannery, Brian P., *Numerical Recipes: The Art of Scientific Computing*, 3rd Edition, Cambridge University Press, September 10, 2007

[24] Press, William H., Teukolsky, Saul A., Vetterling, William T., and Flannery, Brian P., *Numerical Recipes: The Art of Scientific Computing*, 3rd Edition, http://numerical.recipes/

[25] Press, William H., *Opinionated Lessons in Statistics*, http://www.opinionatedlessons.org/

[26] Platt, J., *Probabilistic outputs for support vector machines and comparison to regularized likelihood methods*, Advances in Large Margin Classifiers, Smola, A., Bartlett, P., Scholkopf, B. and Schuurmans, D. (Eds.), Cambridge, MA: MIT Press, 2000

[27] Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Müller, K.-R., Rätsch, G. and Smola, A., *Input Space vs. Feature Space in Kernel-Based Methods*, IEEE Transactions on Neural Networks, Volume 10, Number 5, pp. 1000–1017, 1999

[28] Scholkopf, B.; Smola, A. J., *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*, Cambridge, MA: MIT Press, 2002

[29] Scholkopf, B., Smola, A. and Muller, K.R., *Nonlinear component analysis as a kernel eigenvalue problem*, Neural Computation, Volume 10, pp. 1299–1319, 1998

[30] Specht, D. F., *Probabilistic Neural Networks for Classification, Mapping , or Associative Memory*, IEEE International Conference on Neural Networks, pp. 525-532, 1989

[31] Specht, D. F., *Probabilistic Neural Networks*, In Neural Networks, Volume 3, pp. 109-118, 1990

[32] Tomasi, C., *Estimating Gaussian Mixture Densities with EM – A Tutorial*, Duke University Course Notes, 2006, http://www.cs.duke.edu/courses/spring04/cps196.1/handouts/EM/tomasiEM.pdf, Retrieved Sept 2006

[33] Wand, M. and Jones, M., *Kernel Smoothing*, London: Chapman & Hall, 1995