2017

Local probability based transformations and diagonal projection: a new Support Vector Machine-like method for classification of feature vectors

Charles F. Babbs
Purdue University, babbs@purdue.edu

Local probability based transformations and diagonal projection: a new Support Vector Machine-like method for classification of feature vectors

Charles F. Babbs, MD, PhD

Weldon School of Biomedical Engineering,

Purdue University, West Lafayette, Indiana, USA

Background

Support vector machine (SVM) techniques perform classification of input feature vectors. A classifier takes a vector of feature values that describe an object as its input and assigns a label such as "class A" or "class B" as its output, depending on the particular values of the features in the input vector. In our mammography application the feature vector $\mathbf{x} = (x_1, x_2, \cdots x_n)$ represents a set of properties of a mammogram that can be reduced to individual real numbers, $x_1, x_2, \cdots x_n$. The output is a label class A = abnormal, class B = normal. Classifiers are developed using training data that have known or "ground truth" class labels. During the training process the training data are used to specify the parameters in the logical rules used by the classifier to assign a label. The parameters obtained from training data are incorporated into the classifier that is used for testing.

During subsequent testing class labels are assigned to a series of unknown input vectors. In many classifiers the components of the input vector are plotted in an n-dimensional feature space. The testing rules effectively divide the feature space into regions corresponding to the different class labels. For example, in a two-class classification problem if the input vector plots into certain defined regions of feature space it is given label "A", otherwise it is given label "B".

One approach to making such classifiers more sophisticated is to make the decision surfaces in feature space more complex—for example, curved, budging, convoluted in shape like the fiords of Norway, or even insular and disjointed in shape like the country of Pakistan. The secret to making successful complex decision surfaces is the introduction of conditional rules for different regions of feature space. That is "if **x** falls in this particular region then apply this rule, however if **x** falls in that particular region then apply a different rule". Binary tree classifiers and multi stage cascading classifiers work in this way.

The opposite strategy from sculpting a more complex decision surface, which is used in so-called support vector machine (SVM) approaches, can be called "feature conditioning",

using functions of the original raw features instead of the raw features themselves. The functions change the representation of the data, to warp the feature space in a way that makes for easy classification using a simple linear or planar decision surface in the new feature space.

For example if the raw feature data are denoted $\mathbf{x} = (x_1, x_2, \cdots x_n)$, then a transformed feature vector $\phi(\mathbf{x}) = (\phi_1(x_1), \phi_2(x_2), \cdots \phi_n(x_n))$ could be used to make the raw features better behaved. A whitening transformation is an example of such feature conditioning. The first generation Babbs/Sun power function transformation is another example of such a method. In this case the idea of the power function transformation is to make the standard deviations of all features roughly the same size, so that the overall distribution in hyperspace is roughly spherical or globular (that is, the same diameter in all dimensions).

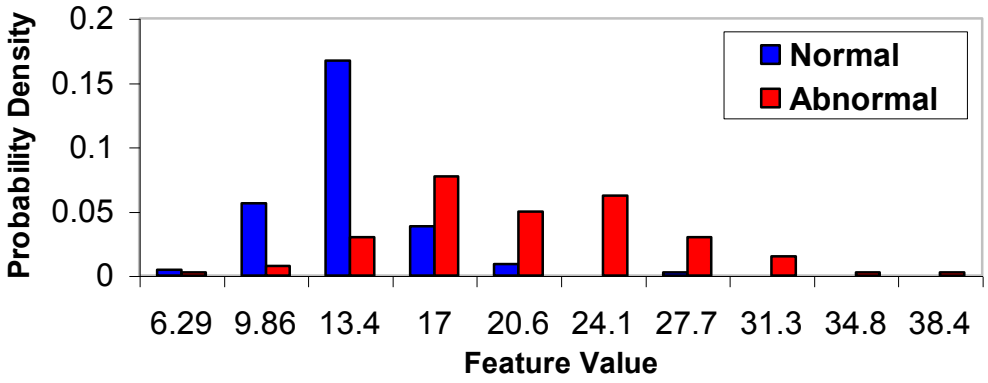
In general, the transformation produces a mapping from the original feature space to a new feature space: $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \cdots \phi_n(\mathbf{x}))$. The arguments of the functions used may include more than one raw feature, as indicated here. Such transformations are helpful when the raw feature distributions for each class are non-globular, interpenetrating, bimodal, or otherwise irregular in shape, in which case they cannot be separated neatly into classes by a simple plane in the original feature space.

We can call the transformed features $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \cdots \phi_n(\mathbf{x}))$ "meta-features". The meta-features are designed to be better behaved, to simplify the subsequent classification task, and to improve its accuracy. For example, a simple plane in the warped feature space may separate classes "A" and "B", which would not have been possible in the original feature space. (The effects are similar to Schwarz-Christoffel transformations in electrical engineering, in which complex electric fields and boundary conditions become simple after deliberate warping of space by functions of complex variables.)

Here we introduce special meta-features based upon probability density functions of the training data. These particular meta-features have particularly interesting properties, as will be seen, that simplify and may improve the accuracy of classification.

Local probability difference transformations

Let us focus first on the training phase, and let us assume that sufficient ground truth training data are available to construct histograms of the distributions of each feature for both known A and known B classes, for example, normal vs. abnormal mammograms. When frequency distributions are divided by the number in each class a "probability density function" or pdf is obtained. The pdf is the frequency of points in each bin of the histogram, divided by the total number points available and the width of the class interval in the histogram. The histograms usually have different shapes and positions for the different for classes A and B. However, the area under each pdf curve is equal to 1. Note that for each feature, x, the pdf_A and pdf_B are functions of individual feature values x.



Define the local probability $p_A(x_i) = \dfrac{pdf_A(x_i)}{pdf_A(x_i) + pdf_B(x_i)}$.

Define the local probability $p_B(x_i) = \dfrac{pdf_B(x_i)}{pdf_A(x_i) + pdf_B(x_i)}$.

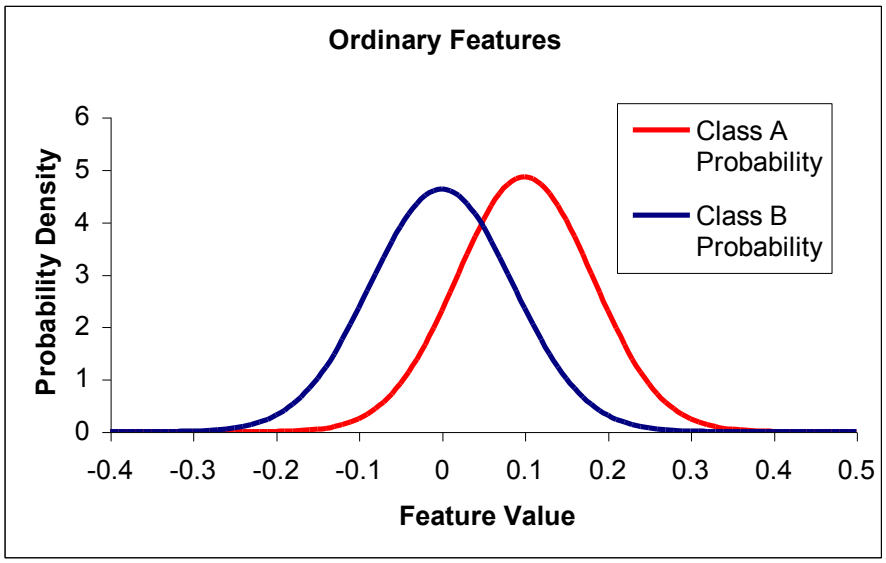
For a two class discrimination problem $p_A(x_i) = 1 - p_B(x_i)$.

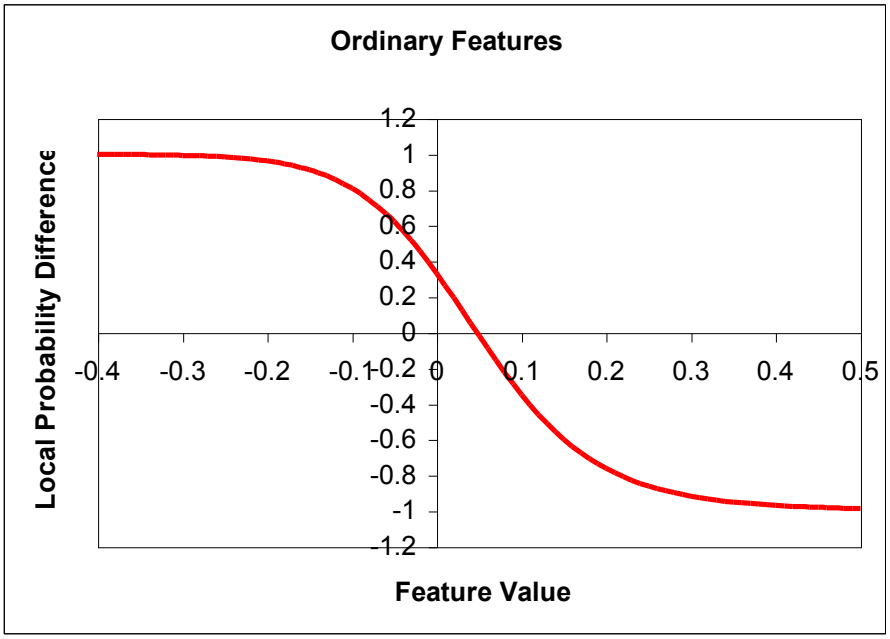
These local probabilities are conditional probabilities. They are the probabilities of an unknown being class A or class B, given a certain value of x. Thus, they tend to capture the same type of information about the relative position of the input vector **x** in feature space, as do sophisticated binary tree or multi-stage classifiers.

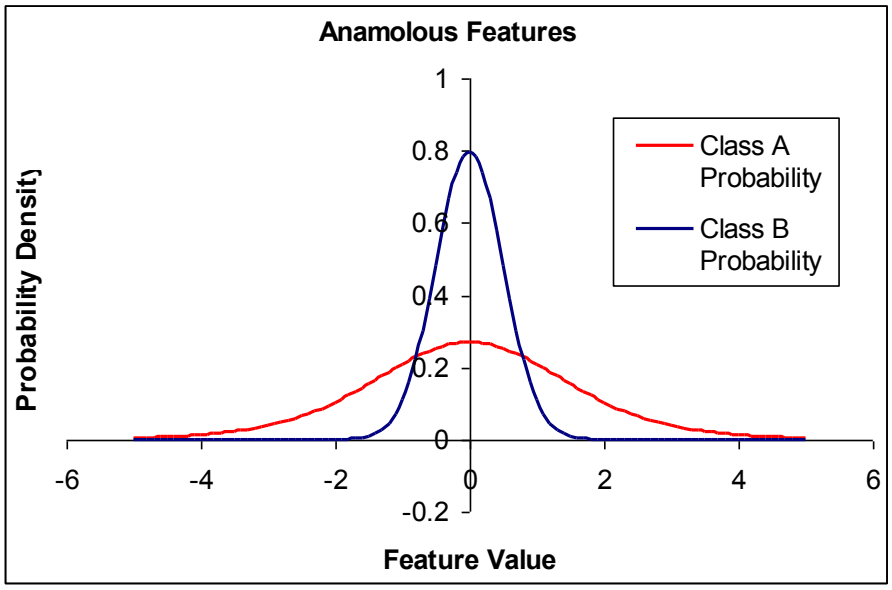
To implement a support vector machine, we seek functions $\phi_1(x_1), \phi_2(x_2), \cdots \phi_n(x_n)$ that are related to which class, A or B, is more likely for a given input, **x**. Here one could use the functions $p_A(x_i)$ directly, which would have values near 1 when class A is likely and values near 0 when class B is likely. However, a particular function with very interesting properties is the local probability difference (LPD), which is defined as follows:

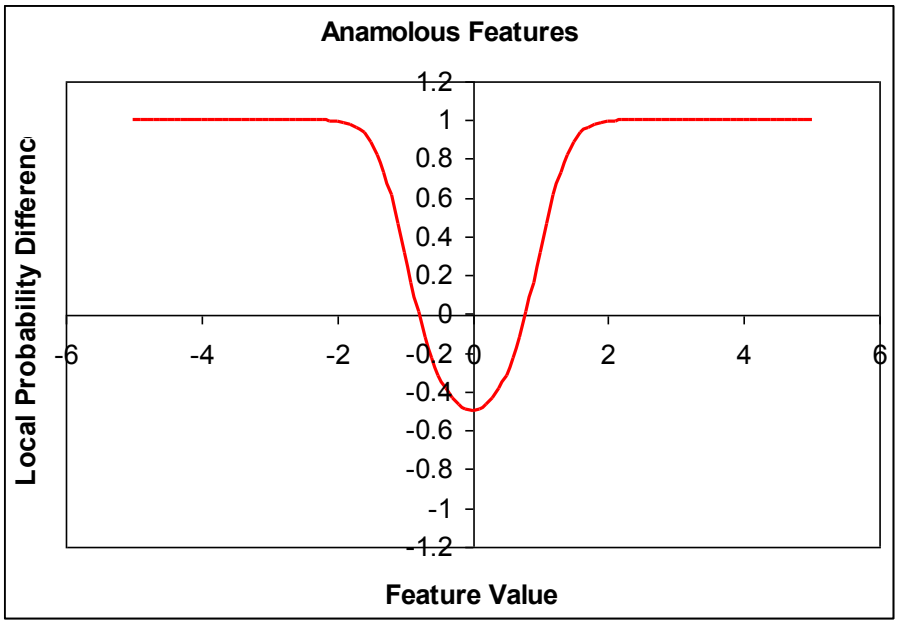
$$\Delta p(x_i) = \frac{pdf_A(x_i) - pdf_B(x_i)}{pdf_A(x_i) + pdf_B(x_i)}.$$

The local probability difference has value 1.0 when class A is almost certain, -1.0 when class B is almost certain, and 0 when A and B are equally likely. The sketches below illustrate the behavior of the LPD function for normally behaved features and for anomalous features in which information is coded in the variance of the distribution rather than the mean.





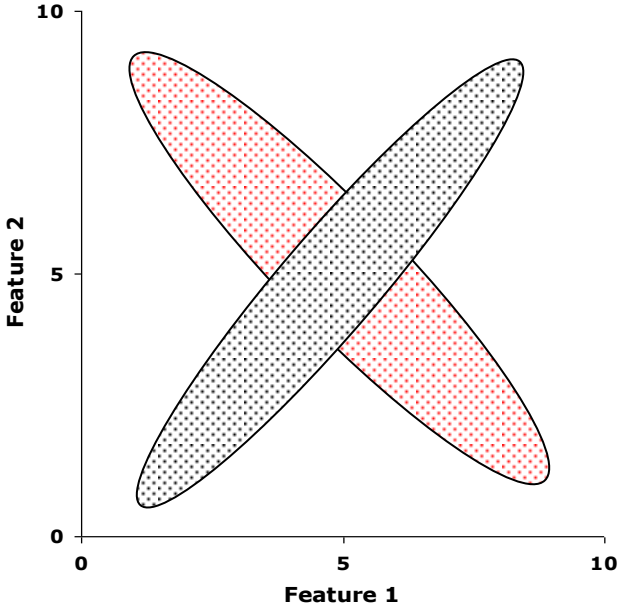

Anamolous Features
Probability Densit)
1
0.8
0.6
0.4
0.2
0
-0.2
-6
-4
-2
0
2
4
6
Feature Value
Class A Probability
Class B Probability


Anamolous Features
Local Probability Differenc
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-6
-4
-2
0
2
4
6
Feature Value

Based upon known training data, a complete vector of local probability difference (LPD) functions can be used to define a new transformed feature space

$$\phi(\mathbf{x}) = (\Delta p_1(x_1), \Delta p_2(x_2), \cdots \Delta p_n(x_n)) .$$

Thus the LPD function captures the essence of the relative density of training points in hyperspace, virtually regardless of the particular geometry of the actual empirical distributions in Euclidian space. In particular, the LPD function captures essential probability information for overlapping and crossed distributions in one or two dimensions. Of 32 possible types of two-dimensional complex distributions that we have already explored, there is only one type—the diagonally crossed distribution—that is poorly represented by the LPD function. Here is a diagonally crossed distribution of two features.



For such diagonally crossed distributions of features x and y, one can use the preliminary transformation z(x,y) = 1 –2x –2y + 4xy to "uncross" classes A and B, reducing x and y to a new one dimensional feature, z , that can be well handled by a subsequent LPD function.

Decision rules using local probability difference transformations

Assume temporarily that we can create $\phi(\mathbf{x}) = (\Delta p_1(x_1), \Delta p_2(x_2), \cdots \Delta p_n(x_n))$ during the training phase of analysis by some sort of curve fitting procedure using the class A and class B training data. (We'll discuss how to get the probability density functions and probability difference functions from the training data in the next section.)

Now given raw input $\mathbf{x}$ and the ability to create $\phi(\mathbf{x}) = (\Delta p_1(x_1), \Delta p_2(x_2), \cdots \Delta p_n(x_n))$, we are ready to plot the conditioned features in hyperspace. This hyperspace has its origin at point (0, 0, 0 … 0) and extends from one extreme point (1, 1, 1, … 1) to the opposite extreme point (-1, -1, -1, … -1).

A provisional choice for a decision plane in hyperspace is a locus of points equidistant from the point of greatest "A-ness" (1, 1, 1, … 1) in one corner of transform space and the point of greatest "B-ness" (-1, -1, -1, … -1) at the diagonally opposite corner of transform space. For a chosen point **x** in the equal distance decision plane is given by

$$\sum_{i=1}^{n} (\Delta p(x_i) - 1)^2 = \sum_{i=1}^{n} (\Delta p(x_i) + 1)^2 .$$

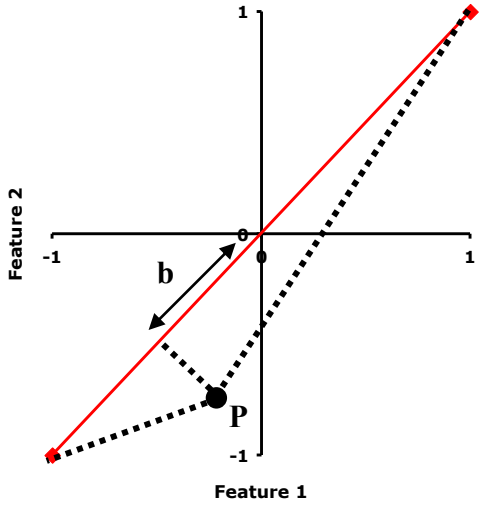
Simplifying,

$$-2\sum_{i=1}^{n} \Delta p(x_i) = 2\sum_{i=1}^{n} \Delta p(x_i), \text{ or } \sum_{i=1}^{n} \Delta p(x_i) = 0 .$$

This is the decision plane! For any given input $\mathbf{x} = (x_1, x_2, \cdots x_n)$, we have

For $\sum_{i=1}^{n} \Delta p(x_i) > 0$, label $\mathbf{x}$ "A", otherwise label $\mathbf{x}$ "B". We just add up the conditioned features. Simple and effective. The simplicity of this decision rule is consistent with the idea that the local probability difference functions reflect an essential quality of the feature distributions related to their separability.

We can produce a range of classifiers with different biases by introducing a constant that represents the distance of the decision plane along the diagonal from (1, 1, 1, … 1) to (-1, -1, -1, … -1).



Let us introduce this constant as a bias term, b, for a decision hyperplane passing through arbitrary point, P, and perpendicular to the diagonal. Now point P is not necessarily equidistant from the two diagonal corners. We see from the Figure that there are two right triangles connected at the corners, joined at distance, b, from the origin along the main diagonal. In hyperspace the distance from the origin (0, 0, 0 … 0) to either corner, (1, 1, 1, … 1) or (-1, -1, -1, … -1), is just $\sqrt{n}$. Hence, by Pythagorean theorem, the shared side of the two right triangles has distance

$$\sum_{i=1}^{n}\left(\Delta p(x_i)-1\right)^2-\left(\sqrt{n}+b\right)^2=\sum_{i=1}^{n}\left(\Delta p(x_i)+1\right)^2-\left(\sqrt{n}-b\right)^2 \text{ , which leads to}$$

$$-4\sum_{i=1}^{n}\Delta p(x_i)=4b\sqrt{n} \quad \text{or} \quad b=\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\Delta p(x_i)$$

(Note this same result can be obtained using the formula for projection of a point in hyperspace onto a line, derived earlier. Details not shown here.)

For a hyperplane passing through either corner point (1, 1, 1, … 1) or (-1, -1, -1, … -1) and perpendicular to the main diagonal, we have, $b=\pm\sqrt{n}$, and $\sum_{i=1}^{n}\Delta p(x_i)=\pm n$,where n is the number of dimensions in feature space. To create an ROC curve, we can scan the decision rule from $\sum_{i=1}^{n}\Delta p(x_i)=-n$ to $\sum_{i=1}^{n}\Delta p(x_i)=+n$ using training data with ground truth labels.

Note that the bias, b, represents the projection of any point P onto the inter-diagonal line, so we can use the simple projection onto this line

$$b = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \Delta p(x_i)$$

to create a one dimensional representation of the data in hyperspace for the purpose of constructing ROC curves. If we re-scale this projected distance as

$$b' = \frac{1}{n} \sum_{i=1}^{n} \Delta p(x_i) \ ,$$

that is, just normalizing by the distance from the origin to a corner, then we have a scale ranging from –1 at one extreme (the class B-most point) through zero (the halfway point) to +1 at the other extreme (the class A-most point). This variable, b', is just the average value of the conditioned features.

In an unbiased mode, b' > 0 generates the label A and b' <= 0 generates the label B. However, any desired decision threshold can be obtained along the ROC curve with -1 < b' < 1. In particular, if class A is considered abnormal, then larger positive values of b' are associated with a larger true positive fraction (TPF) and a smaller false positive fraction (FPF). In a Yajie-like classifier for mammogram analysis one can vary b' to generate desirable low FPF regional classifiers that will be most effective whole image analysis.

Finding transform functions from training data semi-automatically

Background: Mammogram analysis is a form of supervised computer learning in which some rational human intervention is needed during the training phase. In particular the statistical distributions of raw feature data should be scanned by a knowledgeable human for obvious errors, extreme outliers highly unusual shape, etc. In addition, as with the Babbs/Sun power function transformation, it is helpful to cast features into standard form. Here standard form means that feature values are re-centered such that there are no negative values and the smallest positive value in the combined class A or class B distribution is close to zero. For example, the temperature of samples of liquid water would be changed from the Fahrenheit scale (32 – 212) to the centigrade scale (0 to 100), or at least to a scale of "Fahrenheit minus 30" (2 to 180), so that there is no long leader region on the scale with no data. This standard format makes the procedure for curve fitting work more consistently, avoiding ungraceful division by zero and numerical overflow.

In symbols, we define raw training data for classes A and B as x'_A and x'_B . During human supervision of the training process we also define reasonable maximum and minimum values indicating the practical range of each feature. For example if, human body temperature, T, were a feature, which in health and disease ranges from 36 to 43 C in ordinary hospital patients, we might define $T_{min} = 35\,C$ and $T_{max} = 45\,C$. Thus for each distribution of raw feature data we define

$x'_{A\,min}$, $x'_{A\,max}$ $x'_{B\,min}$, and $x'_{B\,max}$ by human inspection of the distributions.

Then we also define $x'_{min} = \min(x'_{A\,min}, x'_{B\,min})$.

The images in standard form are then re-centered as follows for both A and B distributions:

$$x = x' - x'_{min} \ .$$

(For distributions of some of Yajie's features there are many instances of mammograms with raw feature values equal to zero. In this case it is helpful to choose $x'_{min} = \min(x'_{A\,min}, x'_{B\,min})$ as slightly < 0 as an aid to subsequent curve fitting.)

For simplicity of notation, we shall use the unadorned symbol x to represent a feature value in standard form, and a simple bold **x** to represent a feature vector in standard form.

Once features are cast in standard form, having all positive values and no long leader regions, we can implement a variety of strategies to fit curves to the data. Polynomial curve fitting or polynomial regression techniques are flexible and powerful methods, which involve both art and science. There are many possible routes to finding a satisfactory polynomial curve fit. Here two of several possible approaches are described in detail. Polynomial curve fits work well for interpolation between given data points, but they tend to behave wildly extrapolation outside the prescribed range and work very poorly for extrapolation. Human supervision of the curve fitting process is important to prevent evaluating these functions outside their well-behaved range. Here two of several possible approaches are discussed in detail.

Method 1—fitting polynomials to cumulative feature distributions

The first method is to fit polynomial functions to cumulative pdf data, which are less susceptible to noise and sampling variability than probability density data. Then differentiate the cumulative probability density functions to get smooth estimates of actual probability density functions. This strategy helps prevent over-fitting of the training data.

Let $P_A(x) = \int_{-\infty}^{x} pdf_A(x)dx$ and $P_B(x) = \int_{-\infty}^{x} pdf_B(x)dx$.

Also, let x_{Amin} and x_{Amax} be the minimum and maximum values of x for the distribution of class A training data, and let x_{Bmin} and x_{Bmax} be the minimum and maximum values of x for the distribution of class B training data. For these respective ranges of A and B training data one can fit the polynomial functions

$$\hat{P}_A(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = \sum_{i=0}^{n} a_i x^i \quad \text{and}$$

$$\hat{P}_B(x) = b_0 + b_1x + b_2x^2 + \cdots + b_nx^n = \sum_{i=0}^{n} b_i x^i$$

to the training data. (The fitting process will be discussed next.) The probability density functions for the training data can then be found by differentiation:

$$pdf_A(x) = \frac{d\hat{P}_A(x)}{dx} = a_1 + 2a_2x + 3a_3x^2 + \cdots + na_nx^{n-1} = \sum_{i=1}^{n} i\, a_i x^{i-1} \quad \text{for } x_{Amin} \le x \le x_{Amax}$$

and

$$pdf_B(x) = \frac{d\hat{P}_B(x)}{dx} = b_1 + 2b_2x + 3b_3x^2 + \cdots + nb_nx^{n-1} = \sum_{i=1}^{n} i\, b_i x^{i-1} \quad \text{for } x_{Bmin} \le x \le x_{Bmax}.$$

Now define the overall fitted probability density functions

$$\mathrm{pdf}_A(x) = \begin{cases} 0, & x \le x_{A\min} \\ 1, & x \ge x_{A\max} \\ \sum_{i=1}^{n} i\, a_i x^{i-1}, & \text{otherwise} \end{cases} \quad \text{and} \quad \mathrm{pdf}_B(x) = \begin{cases} 0, & x \le x_{B\min} \\ 1, & x \ge x_{B\max} \\ \sum_{i=1}^{n} i\, b_i x^{i-1}, & \text{otherwise} \end{cases}.$$

Once the coefficients of the polynomials are known from analysis of training data, then one can proceed to the testing phase using the local probability difference function for any raw feature value, x_i , as

$$\Delta p(x_i) = \frac{\mathrm{pdf}_A(x_i) - \mathrm{pdf}_B(x_i)}{\mathrm{pdf}_A(x_i) + \mathrm{pdf}_B(x_i)}.$$

To find the set of coefficients a_i and b_i for each feature, we can determine automatically various percentiles of the class A and class B distributions of training data for each feature. A percentile is a value below which a given percentage or fraction of values occur. We denote percentiles by subscripts in percent. For example, the tenth percentile, x_{10} , is the value below which 10 percent of the values fall. In terms of probability density functions a percentile is a value with a specified cumulative frequency, for example

$$0.1 = \int_{-\infty}^{x_{10}} \mathrm{pdf}(x)dx \;\text{ and }\; 0.5 = \int_{-\infty}^{x_{50}} \mathrm{pdf}(x)dx .$$

To perform a polynomial curve fit we first determine a range of percentiles for both class A and class B training data, for example x_0 , x_{10} , x_{30} , x_{50} , x_{70} , x_{90} , x_{100} . The percentiles for the A and B distributions are, of course, different. If we have the complete distribution for the entire population of mammograms, then $x_0 = x_{min}$, and $x_{100} = x_{max}$. If we only have a small sample of training data, then $x_0 > x_{min}$, and $x_{100} < x_{max}$.

These constraints lead to simultaneous linear equations that can be solved for the desired coefficients of the polynomial curve fit, based upon a sample of training data for which various percentiles have been computed. For class A data (the same would be done separately for class B data) we have

$$0 = a_0 + a_1 x_{min} + a_2 x_{min}^2 + \cdots + a_n x_{min}^n$$

$$0.1 = a_0 + a_1 x_{10} + a_2 x_{10}^2 + \cdots + a_n x_{10}^n$$

$$0.3 = a_0 + a_1 x_{30} + a_2 x_{30}^2 + \cdots + a_n x_{30}^n$$

• • •

$$1 = a_0 + a_1 x_{max} + a_2 x_{max}^2 + \cdots + a_n x_{max}^n .$$

In addition, to encourage curve fits having proper sigmoid shape for cumulative probability distributions, we also include the boundary conditions that

$$\frac{d\hat{P}_A(x_{A\,min})}{dx} = a_1 + 2a_2 x_{A\,min} + 3a_3 x_{A\,min}^2 \cdots + n a_n x_{A\,min}^{n-1} = \sum_{i=1}^{n} i\, a_i x_{A\,min}^{i-1} \approx \varepsilon$$

and

$$\frac{d\hat{P}_A(x_{A\,max})}{dx} = a_1 + 2a_2 x_{A\,max} + 3a_3 x_{A\,max}^2 \cdots + n a_n x_{A\,max}^{n-1} = \sum_{i=1}^{n} i\, a_i x_{A\,max}^{i-1} \approx \varepsilon$$

at x_{Amin} and at x_{Amax}, the minimum and maximum values of x for the distribution of class A training data, and similarly at x_{Bmin} and at x_{Bmax}. Here ε is a small value of slope near zero. For a Gaussian-like distributions $\varepsilon \approx 0.05/(x_{99} - x_1)$. This estimate, or zero, can be used to similar effect.

Example: Suppose we want to fit a 7^{th} order polynomial to the data for class A.

$$\hat{P}_A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7$$

subject to the boundary conditions that

$$\hat{P}_A(x_{A\min}) = 0 \ , \quad \hat{P}_A(x_{A\max}) = 1 \ , \quad \frac{d\hat{P}_A(x_{A\min})}{dx} = \varepsilon \ , \text{ and } \quad \frac{d\hat{P}_A(x_{A\max})}{dx} = \varepsilon \ .$$

If we have found the 20^{th}, 40^{th}, 60^{th}, and 80^{th} percentiles for the class A training data, then we have a system of 8 linear equations:

$$a_0 + a_1 x_{A\min} + a_2 x_{A\min}^2 + a_3 x_{A\min}^3 + a_4 x_{A\min}^4 + a_5 x_{A\min}^5 + a_6 x_{A\min}^6 + a_7 x_{A\min}^7 = 0$$

$$a_0 + a_1 x_{20} + a_2 x_{20}^2 + a_3 x_{20}^3 + a_4 x_{20}^4 + a_5 x_{20}^5 + a_6 x_{20}^6 + a_7 x_{20}^7 = 0.20$$

$$a_0 + a_1 x_{40} + a_2 x_{40}^2 + a_3 x_{40}^3 + a_4 x_{40}^4 + a_5 x_{40}^5 + a_6 x_{40}^6 + a_7 x_{40}^7 = 0.40$$

$$a_0 + a_1 x_{60} + a_2 x_{60}^2 + a_3 x_{60}^3 + a_4 x_{60}^4 + a_5 x_{60}^5 + a_6 x_{60}^6 + a_7 x_{60}^7 = 0.60$$

$$a_0 + a_1 x_{80} + a_2 x_{80}^2 + a_3 x_{80}^3 + a_4 x_{80}^4 + a_5 x_{80}^5 + a_6 x_{80}^6 + a_7 x_{20}^7 = 0.80$$

$$a_0 + a_1 x_{A\max} + a_2 x_{A\max}^2 + a_3 x_{A\max}^3 + a_4 x_{A\max}^4 + a_5 x_{A\max}^5 + a_6 x_{A\max}^6 + a_7 x_{A\max}^7 = 1$$

$$0 + a_1 + 2a_2 x_{A\min} + 3a_3 x_{A\min}^2 + 4a_4 x_{A\min}^3 + 5a_5 x_{A\min}^4 + 6a_6 x_{A\min}^5 + 7a_7 x_{A\min}^6 = \varepsilon$$

$$0 + a_1 + 2a_2 x_{A\max} + 3a_3 x_{A\max}^2 + 4a_4 x_{A\max}^3 + 5a_5 x_{A\max}^4 + 6a_6 x_{A\max}^5 + 7a_7 x_{A\max}^6 = \varepsilon$$

or

$$\begin{pmatrix} 1 & x_{\min} & x_{\min}^2 & x_{\min}^3 & x_{\min}^4 & x_{\min}^5 & x_{\min}^6 & x_{\min}^7 \\ 1 & x_{20} & x_{20}^2 & x_{20}^3 & x_{20}^4 & x_{20}^5 & x_{20}^6 & x_{20}^7 \\ 1 & x_{40} & x_{40}^2 & x_{40}^3 & x_{40}^4 & x_{40}^5 & x_{40}^6 & x_{40}^7 \\ 1 & x_{60} & x_{60}^2 & x_{60}^3 & x_{60}^4 & x_{60}^5 & x_{60}^6 & x_{60}^7 \\ 1 & x_{80} & x_{80}^2 & x_{80}^3 & x_{80}^4 & x_{80}^5 & x_{80}^6 & x_{80}^7 \\ 1 & x_{\max} & x_{\max}^2 & x_{\max}^3 & x_{\max}^4 & x_{\max}^5 & x_{\min}^6 & x_{\min}^7 \\ 0 & 1 & 2x_{\min} & 3x_{\min}^2 & 4x_{\min}^3 & 5x_{\min}^4 & 6x_{\min}^5 & 7x_{\min}^6 \\ 0 & 1 & 2x_{\max} & 3x_{\max}^2 & 4x_{\max}^3 & 5x_{\max}^4 & 6x_{\max}^5 & 7x_{\max}^6 \end{pmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0.2 \\ 0.4 \\ 0.6 \\ 0.8 \\ 1.0 \\ \varepsilon \\ \varepsilon \end{Bmatrix}$$

This system can be solved for the constants a_0 through a_8 for each component of the feature vector. This process is repeated using the training data for class B mammograms to obtain the training data-based constants, b_0 through b_8 , for the probability density functions describing the class B distributions.

With the training data-based constants a_i and b_i now available for all features, i , the testing and classification module can be specified to evaluate the local probability difference functions $\Delta p(x)$ for every feature, using

$$pdf_A(x) = a_1 + 2a_2x + 3a_3x^2 + 4a_4x^3 + 5a_5x^4 + 6a_6x^5 + 7a_7x^6 \ ,$$

$$pdf_B(x) = b_1 + 2b_2x + 3b_3x^2 + 4b_4x^3 + 5b_5x^4 + 6b_6x^5 + 7b_7x^6 \ ,$$

and the discriminate function

$$\Delta p(x_i) = \frac{pdf_A(x_i) - pdf_B(x_i)}{pdf_A(x_i) + pdf_B(x_i)}.$$

The following tables and figures show application of the cumulative regression method to two sets of test data. Class A is a normal distribution with mean 3 and standard deviation 1. Class B is a linear combination of equal parts of Class A and a normal distribution with mean 6 and standard deviation 1.

X Class A	X Class B
0.66	0.95
2.16	3
3	4.5
3.84	6
5.34	8.05

The matrices for linear equations and the solutions (here for a 6^{th} order polynomial) are

rhs	matrix -- Class B 6th order						
0.01	1	0.66	0.4356	0.287496	0.189747	0.125233	0.125233
0.2	1	2.16	4.6656	10.0777	21.76782	47.0185	101.56
0.5	1	3	9	27	81	243	729
0.8	1	3.84	14.7456	56.6231	217.4327	834.9416	3206.176
0.99	1	5.34	28.5156	152.2733	813.1394	4342.165	23187.16
0.0106838	0	1	1.32	1.3068	1.149984	0.948737	0.7514
0.0106838	0	1	10.68	85.5468	609.0932	4065.697	26052.99

slope at 1st and 99th percential about 0.01*5 / (x99 -x1)

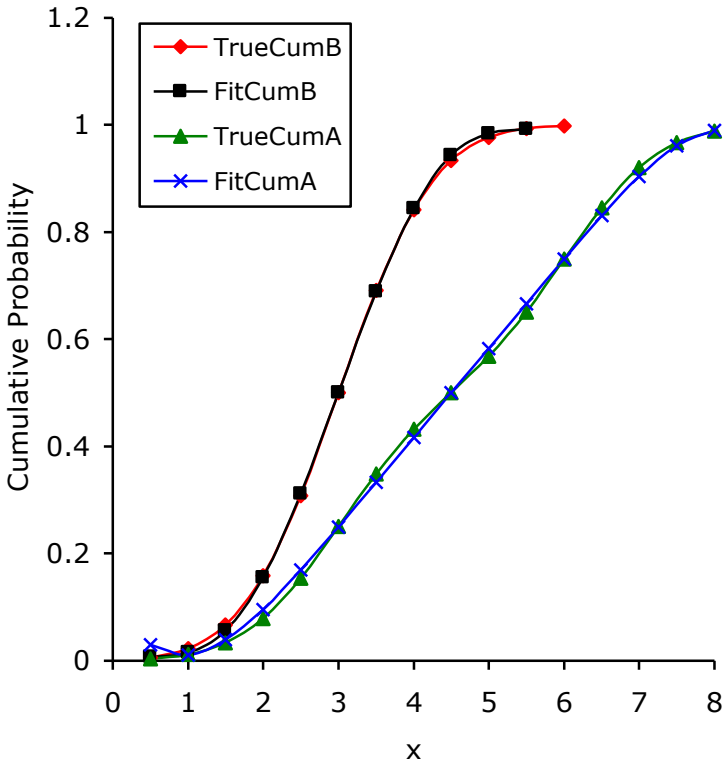
rhs	matrix -- Class A 6th order						
0.01	1	0.95	0.9025	0.857375	0.814506	0.773781	0.773781
0.25	1	3	9	27	81	243	729
0.5	1	4.5	20.25	91.125	410.0625	1845.281	8303.766
0.75	1	6	36	216	1296	7776	46656
0.99	1	8.05	64.8025	521.6601	4199.364	33804.88	272129.3
0.0070423	0	1	1.9	2.7075	3.4295	4.072531	4.642686
0.0070423	0	1	16.1	194.4075	2086.641	20996.82	202829.3

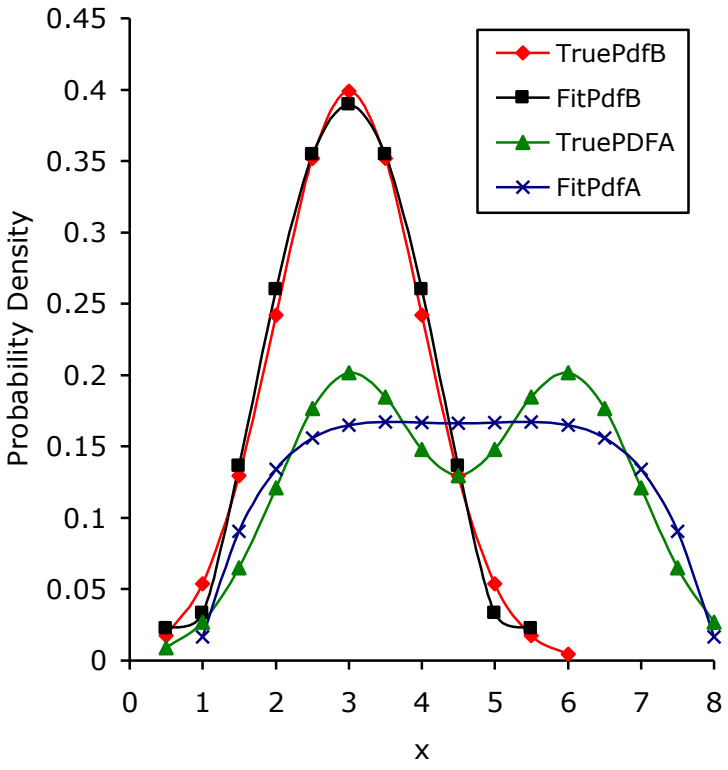
In this example the sample percentiles were taken at percentiles 1, 20, 50, 80, and 99 for class B and at percentiles 1, 25, 50, 75, and 99 for class A. For the bimodal Class A data the percentiles were deliberately chosen for evaluation at the peaks and the trough of the distribution. The heading "rhs" indicates the right hand side values.

This system of linear equations was solved using the Gauss-Jordan method to obtain the following coefficients.

Name	Solutions	Name	Solutions
b_0	-0.03656	a_0	0.119503
b_1	0.193422	a_1	-0.27135
b_2	-0.29771	a_2	0.204208
b_3	0.194428	a_3	-0.04682
b_4	-0.04034	a_4	0.005282
b_5	0.002689	a_5	-0.00023
b_6	-1.1E-17	a_6	-2.1E-18

Results of the curve fit vs. true values (from the standard functions for the normal distribution) are shown below. The probability density functions (pdf(x)) are shown on the next page.





The cumulative curve fitting method is insensitive to individual peaks in the training data, which may be a desirable feature to avoid over-fitting of the training data—a known problem in automatic classification.

Method 2—fitting polynomials directly to histogram data

Suppose that a bin of a histogram contains n counts of total of N training points. If w is the bin width (that is the difference between the upper limit and the lower limit of the bin) and if x is the midpoint of the bin, then pdf(x) = n/(N w).

For a unimodal distribution let x_1, x_2, x_3, x_4, and x_5 represent the midpoints of bins in the histogram of the training data near the tails (1 and 5), the mode (3), and halfway down the slopes (2 and 4). Let p_1, p_2, … p_5 be the corresponding measured pdf values from the bins of the histogram.

If the cumulative probability density is represented as

$$\hat{P}(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_7 x^7 \quad \text{for } x_{min} \le x \le x_{max} \text{ ,and}$$

$$\text{pdf}(x) = \frac{d\hat{P}(x)}{dx} = a_1 + 2a_2 x + 3a_3 x^2 + \cdots + 7a_7 x^6$$

then

$$\text{pdf}'(x) = \frac{d(\text{pdf}(x))}{dx} = 2a_2 + 2 \cdot 3a_3 x + \cdots + 6 \cdot 7a_7 x^5 .$$

Working with pdf and pdf slopes for training data directly, we have the following system of linear equations for unknown coefficients, a_i .

$$\begin{pmatrix} 1 & 2x_1 & 3x_1^2 & 4x_1^3 & 5x_1^4 & 6x_1^5 & 7x_1^6 \\ 1 & 2x_2 & 3x_2^2 & 4x_2^3 & 5x_2^4 & 6x_2^5 & 7x_2^6 \\ 1 & 2x_3 & 3x_3^2 & 4x_3^3 & 5x_3^4 & 6x_3^5 & 7x_3^6 \\ 1 & 2x_4 & 3x_4^2 & 4x_4^3 & 5x_4^4 & 6x_4^5 & 7x_4^6 \\ 1 & 2x_5 & 3x_5^2 & 4x_5^3 & 5x_5^4 & 6x_5^5 & 7x_5^6 \\ 0 & 2 & 6x_1 & 12x_1^2 & 20x_1^3 & 30x_1^4 & 42x_1^5 \\ 0 & 2 & 6x_5 & 12x_5^2 & 20x_5^3 & 30x_5^4 & 42x_5^5 \end{pmatrix} \cdot \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{Bmatrix} = \begin{Bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ +\varepsilon \\ -\varepsilon \end{Bmatrix} ,$$

where for the direct pdf curve fit the value of ε can be zero or $\varepsilon \approx 0.3/(x_{99} - x_1)$.

Using the same test distributions of unimodal and bimodal binomial distributions as before, we have:

Percentiles	rhs	matrix -- Class B 6th order						
1	0.0258	1	1.32	1.3068	1.149984	0.948737	0.7514	0.578578
10	0.1758	1	3.44	8.8752	20.35379	43.76065	90.32199	181.2461
50	0.3989	1	6	27	108	405	1458	5103
90	0.1758	1	8.56	54.9552	313.611	1677.819	8617.278	43028.94
99	0.0258	1	10.68	85.5468	609.0932	4065.697	26052.99	162310.1
slope1	0.064103	0	2	3.96	5.2272	5.74992	5.692421	5.259797
slope99	-0.0641	0	2	32.04	342.1872	3045.466	24394.18	182370.9

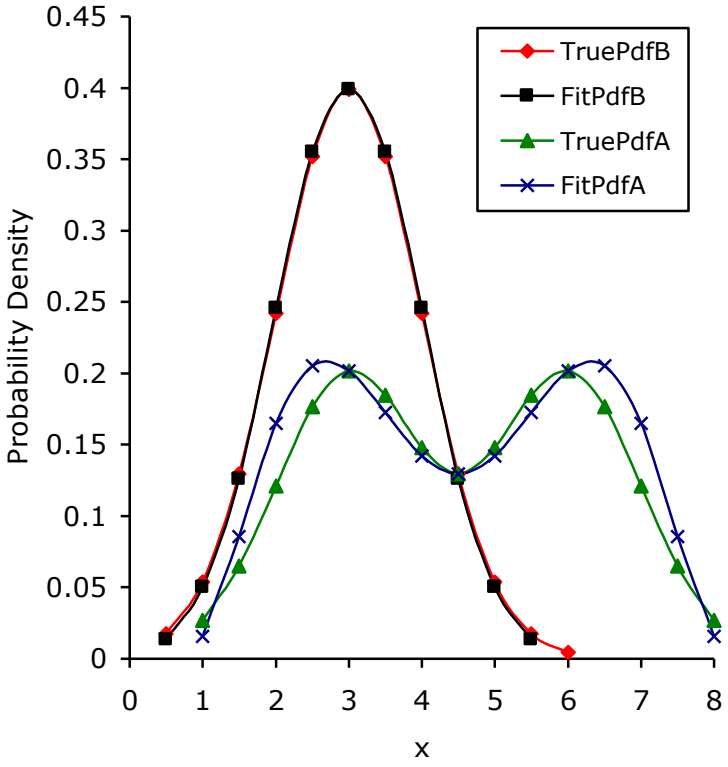
slope at 1st and 99th percential about +- 0.06*5 / (x99 -x1)

	rhs	matrix -- Class A 6th order						
1	0.0129	1	1.9	2.7075	3.4295	4.072531	4.642686	5.145643
26	0.20165	1	6	27	108	405	1458	5103
50	0.1295	1	9	60.75	364.5	2050.313	11071.69	58126.36
74	0.20165	1	12	108	864	6480	46656	326592
99	0.0129	1	16.1	194.4075	2086.641	20996.82	202829.3	1904905
slope1	0.042254	0	2	5.7	10.83	17.1475	24.43519	32.4988
slope99	-0.04225	0	2	48.3	777.63	10433.2	125980.9	1419805

and solutions

Name	Solutions	Name	Solutions
b_1	-0.12583	a_1	0.469255
b_2	0.292666	a_2	-0.67197
b_3	-0.30402	a_3	0.451127
b_4	0.177455	a_4	-0.14177
b_5	-0.04836	a_5	0.022939
b_6	0.006088	a_6	-0.00185
b_7	-0.00029	a_7	5.87E-05

with the graphical results of the curve fit on the following page.



For the bimodal distribution (Class A) we assigned x_1 and x_5 to the tails, x_2 and x_4 to the first and second peaks, and x_3 to the nadir between peaks. Then we proceeded as before for unimodal Class B training data.

This method tracks the individual peaks of the empirical pdf data better.

Thus we have come full circle in creating a new support vector machine-like classifier. The general paradigm of the support vector machine is to convert the raw feature vector $\mathbf{x} = (x_1, x_2, \cdots x_n)$ into a transformed feature vector $\phi(\mathbf{x}) = (\phi_1(x_1), \phi_2(x_2), \cdots \phi_n(x_n))$, which can be easily classified by a plane in n-dimensional space. In our case we have chosen $\phi(\mathbf{x}) = (\Delta p_1(x_1), \Delta p_2(x_2), \cdots \Delta p_n(x_n))$. Our decision plane is defined simply as $\sum_{i=1}^{n} \Delta p(x_i) =$ a constant such as zero. In the case of pairs of "crossed" feature distributions which are individually not separable but are negatively correlated, then a transformation of the form $z = 1 - 2x - 2y + 4xy$ can be done to convert the initially useless features of the crossed pair into one useful ordinary feature. This new hybrid feature can be treated in the same way as other ordinary features.

Remarks

Remark 1

In dealing with multiple features, some features are strong, that is, highly discriminating, and some features are weak, or poorly discriminating. For real feature data, often both kinds of features have an equal amount of noise. One problem with linear classifiers in high dimensional feature space is that adding extra weaker features tends to add noise but not much discriminating signal. As a result the overall signal-to-noise ratio of the linear classifier can decrease as more features are added, limiting peak performance.

The local probability difference function

$$\Delta p(x_i) = \frac{pdf_A(x_i) - pdf_B(x_i)}{pdf_A(x_i) + pdf_B(x_i)}$$

has a small value for weaker features in regions of overlap between class A and class B distributions. For a constant amount of noise in the individual input feature x, the amount of noise reflected in $\Delta p(x)$ is less for poorly discriminating regions of large overlap in the domain of x. Thus adding weaker features does not necessarily degrade the signal-to-noise ratio of the diagonal projection $b' = \frac{1}{n}\sum_{i=1}^{n} \Delta p(x_i)$ as much as would occur with an ordinary linear classifier. Hence the overall power of the classifier can continue to increase with the number of added features.

Remark 2

If feature x_1 is perfectly correlated with feature x_2, then the function $\Delta p(x_1)$ will be perfectly correlated with $\Delta p(x_2)$. From the form of

$$b' = \frac{1}{n}\sum_{i=1}^{n} \Delta p(x_i)$$

the correlation will merely increase the weight or contribution of the redundant feature x_2. This effect is not especially harmful to performance of the classifier, except that it tends to dilute the contribution of the other features, making them less effective. That is to say, correlation makes the overall classifier behave a little more like a single-feature classifier. For this reason correlated features are less desirable. The best features have high intrinsic separability and are poorly correlated.

Remark 3

Because the diagonal projection $b' = \frac{1}{n}\sum_{i=1}^{n} \Delta p(x_i)$ is a simple average, we have the possibility of deliberately introducing a weighted average,

$$b' = \frac{1}{W}\sum_{i=1}^{n} w_i \Delta p(x_i)$$

where W is the sum of the weights w_i . This makes some features more equal than others. One could examine the effects of different weighting schemes on the overall results of classification. For example, one could take sets of correlated features and adjust weights within each set so that the weights of the separate sets are about equal. This is one strategy for dealing with correlated features. One could also give stronger features more weight than weaker features. Note that the Δp transformation automatically gives lesser weight to features that have little intrinsic separability and even to regions of feature space where there is less separability.

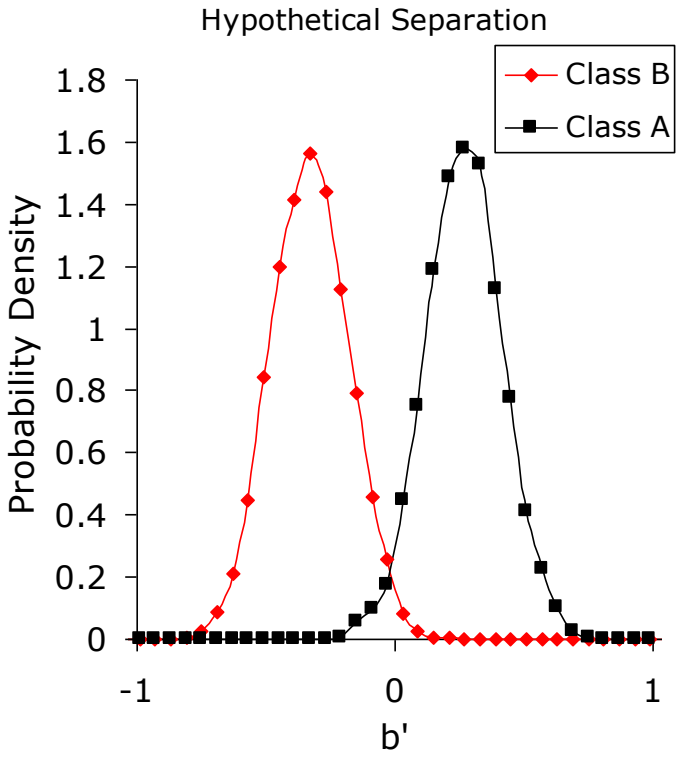
Remark 4

Let us regard $b' = \frac{1}{n}\sum_{i=1}^{n} \Delta p(x_i)$ as a sum of random variables. If we lump together the highly correlated ones, then we can use the statistical properties of the features to predict the statistical properties of b'. Such predictions are useful for validation of computer code and as a tool designing classifiers with a particular theoretical behavior.

Note that for all features the variance of $\Delta p(x)$, denoted $V(\Delta p)$ is < 1, since the maximum expected value of $(\Delta p)^2$ is 1. For "good" features we can say $E\left[(\Delta p)^2\right] = V(\Delta p) \approx 1$. If these features (or internally correlated feature groups) are independent, then the variance

$V(b') \approx \frac{1}{n^2}\sum_{i=1}^{n} 1 = \frac{1}{n}$ and the standard deviation $s(b') \approx \frac{1}{\sqrt{n}}$.

Because of the central limit theorem of statistics, the weighted sums b'_A and b'_B will tend to have a normal distribution in shape as the number of features increases, even though the individual distributions of the Δp variables tend to be more like exponential distributions in shape. Hence the separation of the classes in the b' domain looks like this:



If added new features are independent and have a typical amount of separation, increasing the number of features averaged together will decrease the spread of the two distributions and greatly improve the frequency of correct classification, roughly as a function of $\sqrt{n}$.