Machine Learning: Algorithms and Applications

Philip O. Ogúnbọna

Advanced Multimedia Research Lab
University of Wollongong

Pattern Classification
Autumn

Outline of Topics

What is pattern recognition/classifier?

- We take for granted the fact that we are able to move around in our world and recognize:
  - cars
  - people
  - animals
  - objects in general

  despite the variety in their form and existence.
- What features help in these tasks?
- In pattern recognition we study how to design machines that can recognize and classify ”things”.
- We study the statistics of the features that describe ”things”.
- We study how to measure the performance of pattern recognition systems and select good systems.

Basic Model - Pattern Classifier

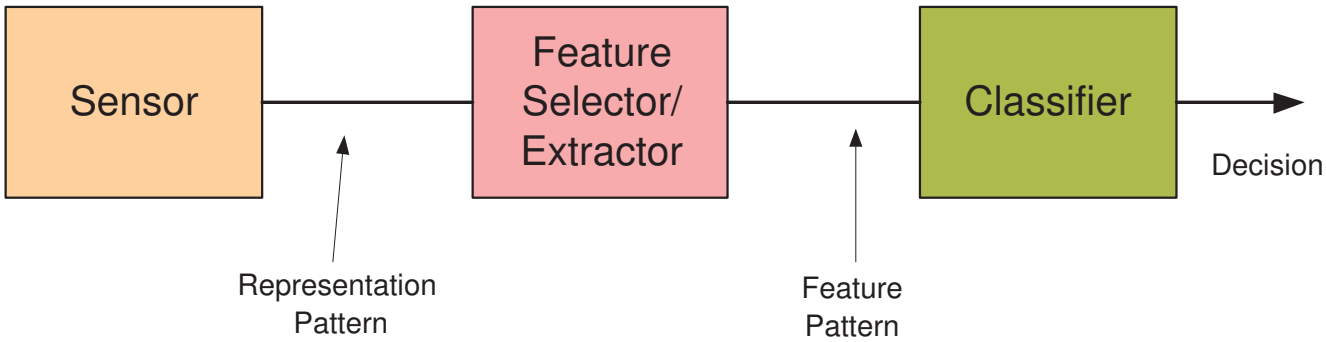


Figure 1: Pattern Classifier Webb (2002)

- The pattern is a set of numbers or values represented as a $p-$ dimensional vector,

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_p \end{bmatrix}^t$$

where t (or sometimes T) denotes vector transpose

Basic Model - Pattern Classifier

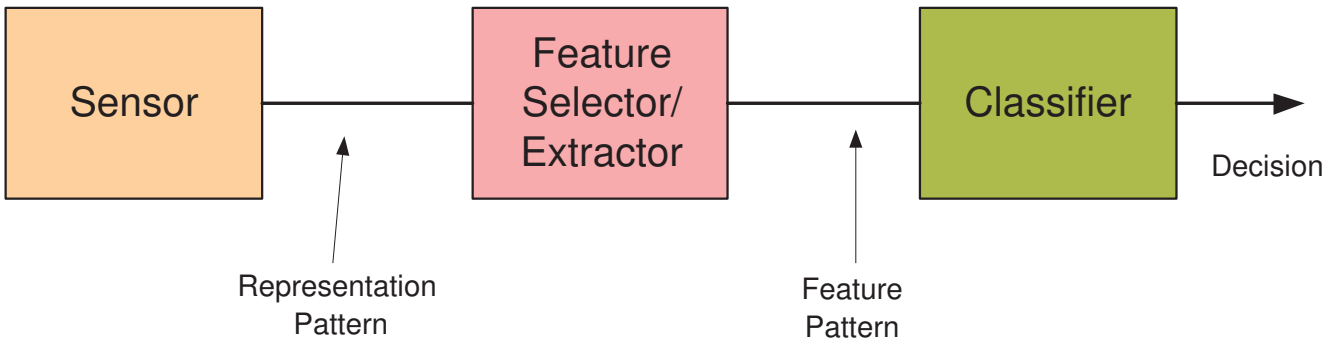


Figure 1: Pattern Classifier Webb (2002)

- The pattern could be:
  - pixels in an image
  - closing prices of a share on the stock market
  - recordings of a speech signal
  - measurements on weather variables
  - group of measurements about a real estate
  - group of measurements about the behaviour and life style of people
  - etc.

Basic Model - Pattern Classifier

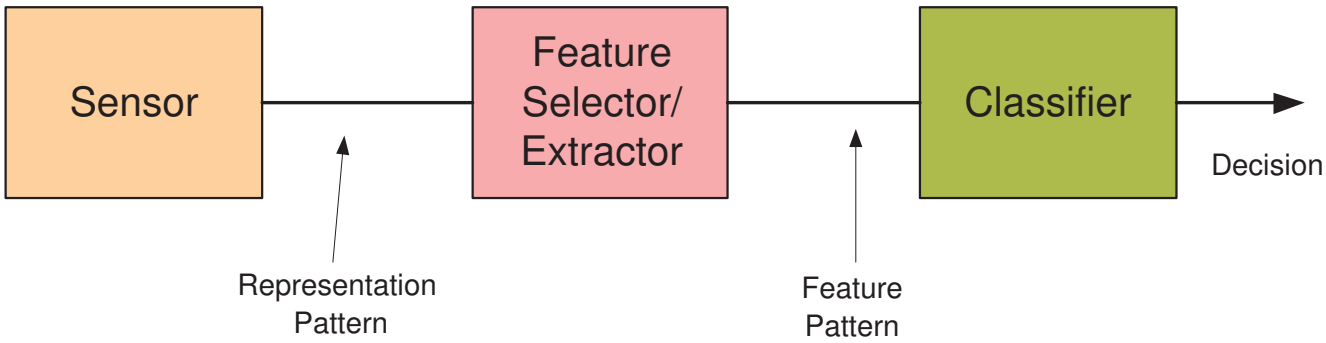


Figure 1: Pattern Classifier Webb (2002)

- We assume that there are $\mathcal{C}$ classes denoted by,

$$\omega_1, \ldots, \omega_{\mathcal{C}}$$

- There is a variable, z, that indicates which class, ω_i, a pattern x belongs. That is,

$$\text{if} \quad z = i, \text{then the pattern, } x, \text{ belongs to } \omega_i, \quad i \in \{1, \ldots, C\}$$

Basic Model - Pattern Classifier

- The problem is how to design the pattern classifier.
- Designing a pattern classifier entails:
  - specifying the classifier model parameters
  - ensuring that response for a given pattern is optimal
- The design process assumes we have a set of patterns of known class, $\{(\mathbf{x}_i, z_i)\}$, called the training or design set, used to design the classifier.
- Part of the design process is to evaluate and set optimal operating parameters.
- The idea is that once we have a designed classifier we can estimate the class membership of an unknown pattern.
- There is an assumption that the samples used for training are drawn from the same probability distribution as the test samples and the operational samples.

Basic Model - Pattern Classifier

- A closer look at the simplified classifier model:

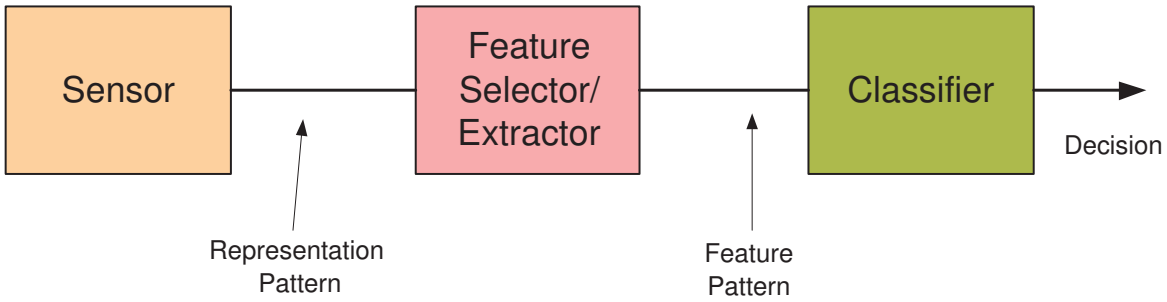


Figure 2: Pattern Classifier Webb (2002)

1. Representation pattern is the raw data we obtain from the sensor e.g. image or video pixels, price of stock, etc.
2. Feature pattern is a small set of variables obtained through some transformation - feature selection and/or extraction
3. The trained classifier uses the feature pattern to make a decision regarding the pattern presented at its input.

Basic Model - Pattern Classifier

- Further consideration about classifier design:

Problem

Given a training set of patterns of known class, we seek to design a classifier that is optimal for the expected operating conditions.

1. The given set of training patterns is *finite*.
2. The classifier model cannot be too complex. In other words it cannot have too many parameters. This situation may lead to over-fitting.
3. It is not important to achieve optimal performance on the design set.
4. It is very important to achieve optimal generalization performance.
   - Expected performance on data representative of the true operating condition - the infinite set from which the design set is drawn.

Supervised and Unsupervised Classification

There are two main categories of classifiers :

Supervised: The classifier design process has a set of data samples with associated labels (class type). These are exemplars or training data.

Unsupervised: The given data is not labelled and the idea is to find groups in the data and the features that distinguish one group from another.

- There is a third category, namely semi-supervised classifiers, in which both labelled and unlabelled data are used for the training.

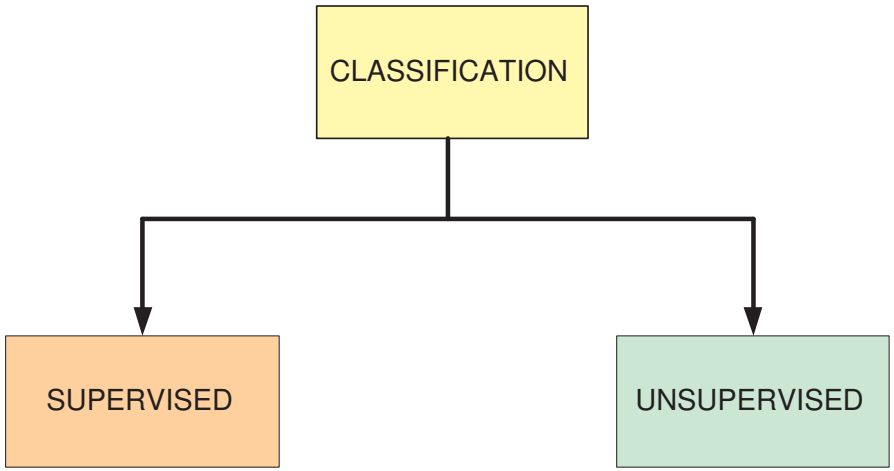


Figure 3: Main categories of classifiers

Supervised Classification

Example - from Duda-Hart-Stork Duda et al. (2001)

We are required to design a classifier for a fishing company so as to automate the sorting process. The company is interested in sorting salmon from bass. The cost of selling a salmon as bass could be high when misclassified!

Salmon

Bass

- Possible features of interest
  - length
  - width
  - number and shape of fins
  - position of mouth
  - lightness
- These features will vary because of measurement errors or conditions.

Supervised Classification

- Interesting observation may show that bass is usually longer than salmon.
- We take several samples of the two fishes and measure their lengths.
- We may represent our measurement as a histogram.
- We may ask the question, "Will this feature sufficiently classify the fishes?"

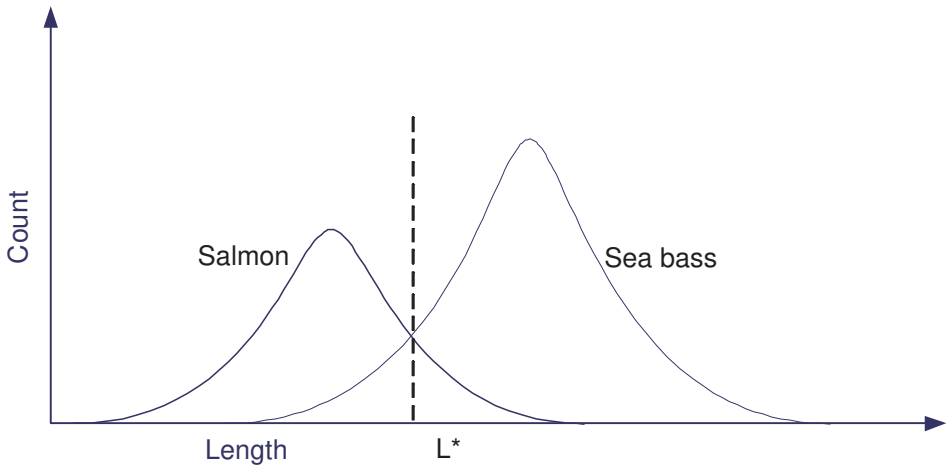


Figure 4: "Histogram" of fishes lengths; Length marked L^* will lead to smallest number of errors.

Supervised Classification

- Perhaps the cost of using *length* alone to classify is too high.
- We take several samples of the two fishes and measure their *lightness*.
- We again represent our measurement as a histogram.
- The answer to the question, "Will this feature sufficiently classify the fishes?" is more satisfying.
- The X^* or L^* is a decision threshold.

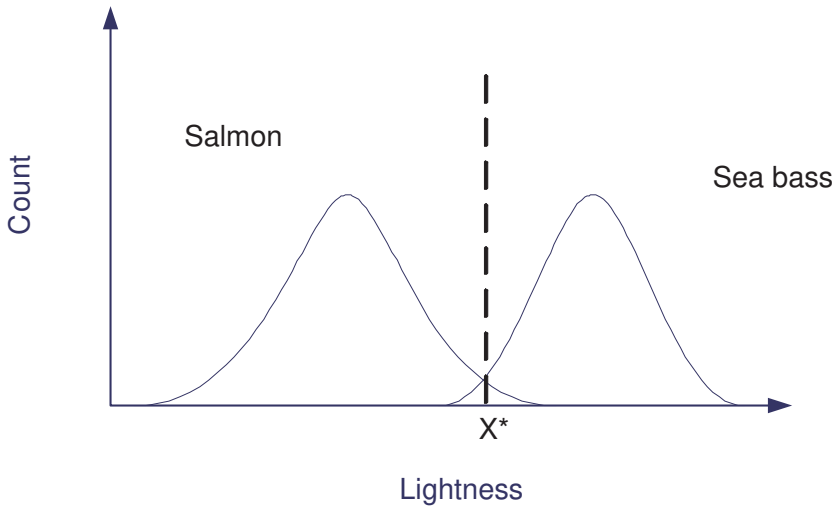


Figure 5: "Histogram" of lightness of fishes; Lightness marked X^* will lead to smallest number of errors.

Supervised Classification

- Assume that we believe that we could do better at classifying the fishes by using two (2) features.
- We now have a two-dimensional feature vector,

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- The feature space can be visualized.
- How to obtain the "best" decision boundary is the classifier design problem.

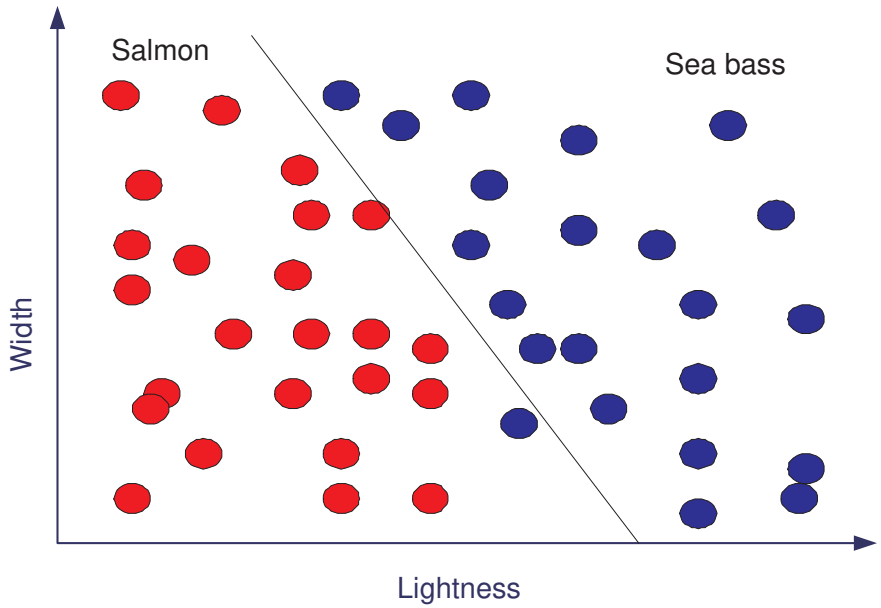


Figure 6: Feature space with decision boundary of classifier.

Supervised Classification

- As we increase the number of features there is need to deal with a high dimensional feature vector.
- The problem of “too many features" is referred to as dimensionality curse.
- A very complicated model may also result in over fitting - training data is separated “perfectly”; new patterns are poorly classified. Generalization problem.

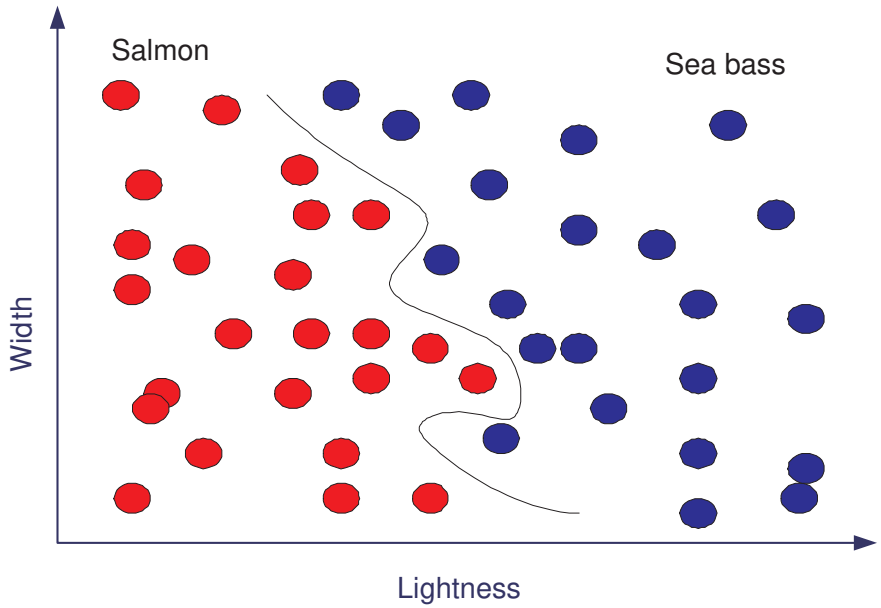


Figure 7: Feature space with complex decision boundary of classifier.

Supervised Classification

Occam's Razor

The principle of using as simple as is necessary model to describe systems is captured in the so-called "Occam's razor" - favour simpler explanations over those that are needlessly complicated.

- The principle underlies the very popular method of sparse representation.

Bayes decision rule - minimum error

- This approach to classification (also called discrimination) assumes that we have full knowledge of the probability density function of each class
- Let the $\mathcal{C}$ classes have known *a priori* probabilities, $P(\omega_1), \ldots, P(\omega_C)$
- We make use of the measurement vector x to assign x to one of the C classes
- Formulate a decision rule to assign x to class ω_j if the probability of class ω_j given the observation x, (i.e. $P(\omega_j|x)$ - posterior probability), is the highest over all classes $\omega_1, \ldots, \omega_C$;

$x \in \omega_j$ if,

$$P(\omega_j|x) > P(\omega_k|x) \quad k = 1, \ldots, C; \; k \neq j$$

- Measurement space is partitioned into $\mathcal{C}$ regions, $\Omega_1, \Omega_2, \ldots, \Omega_C$; $\mathbf{x} \in \Omega_j \Rightarrow \mathbf{x}$ is in class ω_j

Bayes decision rule - minimum error

- We use Bayes' theorem to express the *a posteriori* probabilities $P(\omega_j|x)$ in terms of the *a priori* probabilities and the *class-conditional density functions* $p(x|\omega_i)$

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)}$$

where

$$p(x) = \sum_{j=1}^{C} p(x|\omega_j)P(\omega_j)$$

- In terms of the class-conditional density we can write the decision rule as, assign x to ω_j if,

$$p(x|\omega_j)P(\omega_j) > p(x|\omega_k)P(\omega_k) \quad k = 1, \ldots, C \; k \neq j$$

This is the Bayes' rule for minimum error.

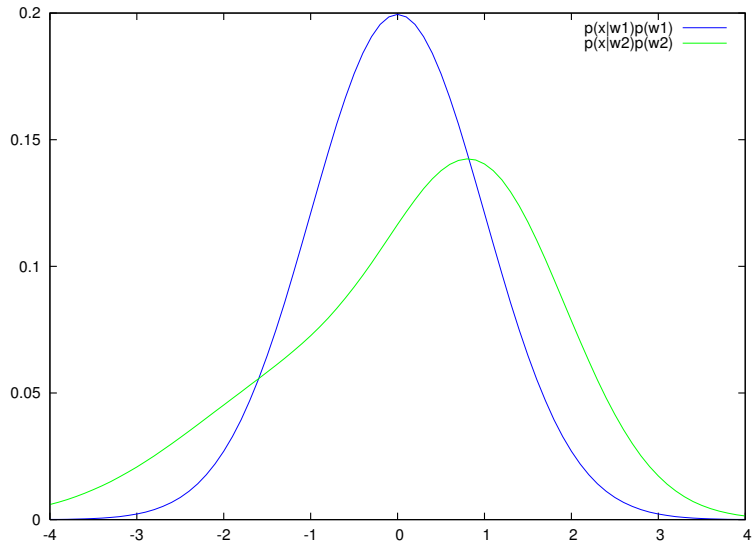
Bayes decision rule - minimum error

- In a two-class case we can write the Bayes' minimum error rule in terms of likelihood ratio, $L_r(x)$, for $x \in$ class ω_1,

$$L_r(x) = \frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)}$$

- Take as an example a two-class discrimination problem, with class ω_1 normally distributed as, $p(x|\omega_1) = N(x|0,1)$ and class ω_2 as a normal mixture with $p(x|\omega_2) = 0.6N(x|1,1) + 0.4N(x|-1,2)$

- The plots of $p(x|\omega_i)p(\omega_i),\ i = 1,2$ with $P(\omega_1) = P(\omega_2) = 0.5$ are shown below;



Bayes decision rule - minimum error

Plots of the likelihood ratio $L_r(x)$ and threshold, $P(\omega_2)/P(\omega_1)$ are shown below:

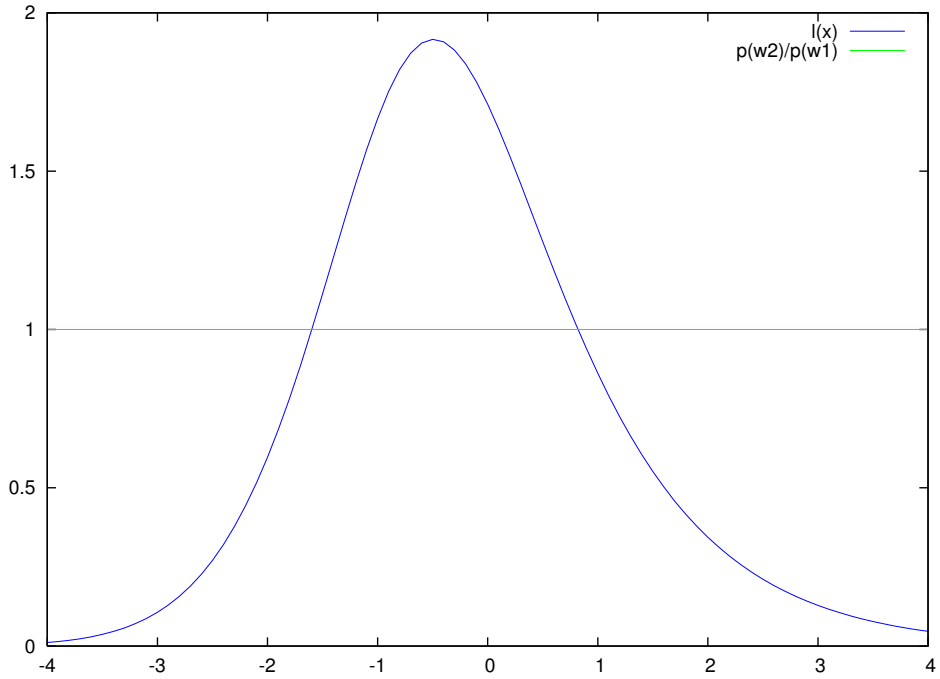


Figure 8: Likelihood function

If $L_r(x) > \frac{P(\omega_2)}{P(\omega_1)}$, the observed sample is classified as ω_1,

Bayes decision rule - minimum risk

- This decision rule minimizes an expected loss or risk.
- Define a loss matrix, Λ, with components,

$$\lambda_{ji} = \text{cost of assigning a pattern } x \text{ to } \omega_i \text{ when } x \in \omega_j$$

- The conditional risk of assigning a pattern x to class ω_i is defined as

$$l_i(x) = \sum_{j=1}^{C} \lambda_{ji} P(\omega_j|x)$$

- The average risk over decision region Ω_i is

$$r_i = \int_{\Omega_i} l_i(x) p(x) dx$$

$$= \int_{\Omega_i} \sum_{j=1}^{C} \lambda_{ji} P(\omega_j|x) p(x) dx$$

Bayes decision rule - minimum risk

- The overall expected cost or risk is obtained by summing the risks associated with all the classes

$$r = \sum_{i=1}^{C} r_i = \sum_{i=1}^{C} \int_{\Omega_i} \sum_{j=1}^{C} \lambda_{ji} P(\omega_j|x) p(x) dx$$

- The risk is minimized if the regions Ω_i are chosen such that if

$$\sum_{j=1}^{C} \lambda_{ji} P(\omega_j|x) p(x) \leq \sum_{j=1}^{C} \lambda_{jk} P(\omega_j|x) p(x) \quad k = 1, \ldots, C$$

  then $x \in \Omega_i$.

- This is the Bayes decision rule for minimum risk

- The Bayes risk, r^*, is

$$r^* = \int_x \min_{i=1,\ldots,C} \sum_{j=1}^{C} \lambda_{ji} P(\omega_j|x) p(x) dx$$

Bayes decision rule - minimum risk

- For a two-category classification problem we can write the conditional risks as:

$$l_i(x) = \sum_{j=1}^{C} \lambda_{ji} P(\omega_j|x)$$

$$l_1(x) = \lambda_{11} P(\omega_1|x) + \lambda_{21} P(\omega_2|x)$$
$$l_2(x) = \lambda_{12} P(\omega_1|x) + \lambda_{22} P(\omega_2|x)$$

- The minimum risk decision rule is simply to decide ω_1 if $l_1(x) < l_2(x)$.
- This can be expressed in terms of posterior probabilities as: Decide ω_1 if

$$(\lambda_{11} - \lambda_{12}) P(\omega_1|x) < (\lambda_{22} - \lambda_{21}) P(\omega_2|x)$$

- In terms of the prior probabilities and conditional densities we decide ω_1 if,

$$(\lambda_{11} - \lambda_{12}) p(x|\omega_1) P(\omega_1) < (\lambda_{22} - \lambda_{21}) p(x|\omega_2) P(\omega_2)$$

Bayes decision rule - minimum risk

- If we consider the special case of equal cost (also called symmetrical or zero-one) loss matrix, Λ, in which,

$$\lambda_{ij} = \begin{cases} 1, & i \neq j; \\ 0, & i = j \end{cases}$$

a substitution of this condition into the Bayes decision rule for minimum risk, gives,

$$\sum_{j=1}^{C} P(\omega_j|x)p(x) - P(\omega_i|x)p(x) \leq \sum_{j=1}^{C} P(\omega_j|x)p(x) - P(\omega_k|x)p(x)$$

for $k = 1, \ldots, C$. This is easily simplified as,

$$p(x|\omega_i)p(\omega_i) \geq p(x|\omega_k)p(\omega_k), \ k = 1, \ldots, C$$

when $x \in$ class ω_i.
This is the same as Bayes rule for minimum error.

Bayes decision rule - minimum risk

- If we consider the special case of zero-one loss matrix, Λ, in which,

$$\lambda_{ij} = \begin{cases} 1, & i \neq j; \\ 0, & i = j \end{cases}$$

and a two-category classification the Bayes decision rule for minimum risk, gives,

$$p(x|\omega_1)p(\omega_1) \geq p(x|\omega_2)p(\omega_2),$$

when $x \in$ class ω_1.
This is the same as Bayes rule for minimum error in the two-category case.

- The corresponding risks in the case of the zero-one loss matrix are

$$\begin{aligned} l_i(x) &= \sum_{j=1}^{C} \lambda_{ij} P(\omega_j|x) \\ &= \sum_{j \neq i} P(\omega_j|x) \\ &= 1 - P(\omega_i|x) \end{aligned}$$

Discriminant Functions

- Bayes decision rules requires knowledge of prior class probabilities and class conditional densities which are often not available in practice and must be estimated from data
- The class of techniques being introduced makes no assumption about $p(x|\omega_i)$ but rather assumes a form of the discriminant functions
- A discriminant function is a function of the feature vector x that leads to a classification rule
- Consider a two-class problem, a discriminant function $h(x)$ is such that

$$\begin{aligned} h(x) > k &\Rightarrow x \in \omega_1 \\ < k &\Rightarrow x \in \omega_2 \end{aligned}$$

for a constant k

Discriminant Functions

- Discriminant functions are not unique. If $f(.)$ is a monotonic function, then

$$g(x) = f(h(x)) > k' \Rightarrow x \in \omega_1$$
$$< k' \Rightarrow x \in \omega_2$$

  where $k' = f(k)$, gives the same decision as $h(x)$.

- For classification problem with C classes we define C discriminant functions, $g_i(x)$ such that,

$$g_i(x) > g_j(x) \Rightarrow x \in \omega_i \quad j = 1, \ldots, C;\ j \neq i$$

  This implies that a feature vector is assigned to the class with the largest discriminant.

- The discriminant techniques rely on the form of the function being specified and not on the underlying distribution

- Parameters of the functional form are adjusted by a training procedure

Linear discriminant functions

- Linear discriminant functions are a linear combination of the components of the measurement (or feature) vector, $\mathbf{x} = \left[x_1, x_2, \ldots, x_p\right]^t$, such that,

$$g(x) = \omega^t x + \omega_0 = \sum_{i=1}^{p} \omega_i x_i + \omega_0$$

  where we need to specify the *weight vector* ω and *threshold weight* ω_0

- The equation describes a *hyperplane* with unit normal in the direction of ω and a perpendicular distance, $|\omega_0|/|\omega|$ from origin.

Linear discriminant functions

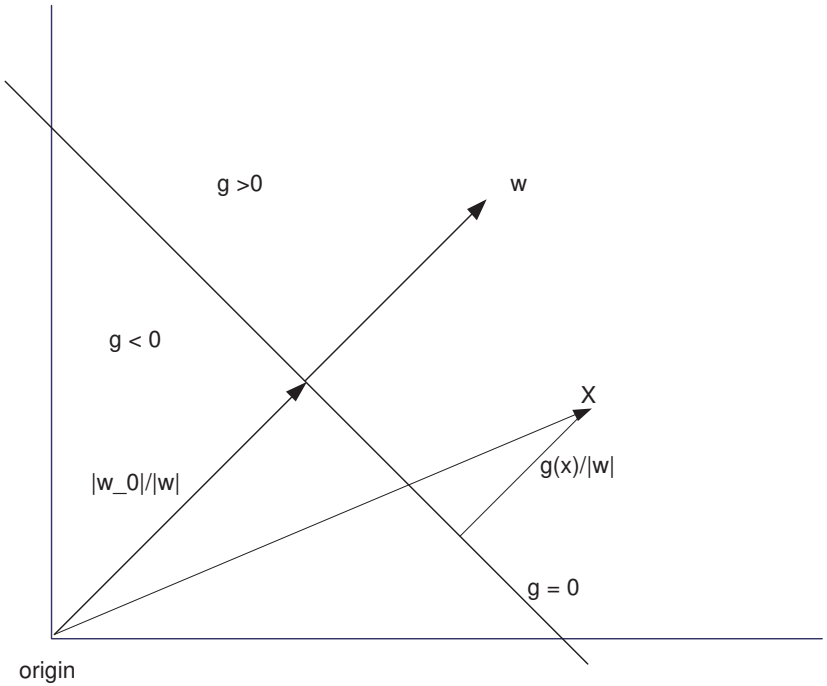


Figure 9: Geometry of linear discriminant function

- The value of the discriminant function for a pattern x is the perpendicular distance from the hyperplane

Linear discriminant functions

- Classifiers that use a linear discriminant function are called linear machine.
- The minimum-distance classifier is an example. It uses the nearest-neighbour decision rule.
- Let the prototype points of the classifier be $p_1, \ldots, p_C$. Each point represents a class, ω_i. The minimum distance classifier assigns x to the ω_i with nearest point p_i

$$||x - p_i||^2 = x^t x - 2x^t p_i + p_i^t p_i$$

The class assigned to x is

$$\omega_i = \max_i (x^t p_i - \frac{1}{2} p_i^t p_i)$$

Linear discriminant functions

- We can relate this assignment to the linear discriminant function

$$g_i(x) = \omega_i^t x + \omega_{i0}$$

where

$$\omega_i = p_i \tag{1}$$

$$\omega_{i0} = -\frac{1}{2}||p_i||^2 \tag{2}$$

to show that it is indeed a linear machine.

- The prototype points could be chosen as the mean of each class and we have a nearest class mean classifier.

Linear discriminant functions

- Each boundary is the perpendicular bisector of the lines joining the prototype points of regions that are contiguous.
- Note also that the decision regions of a linear machine are always convex.

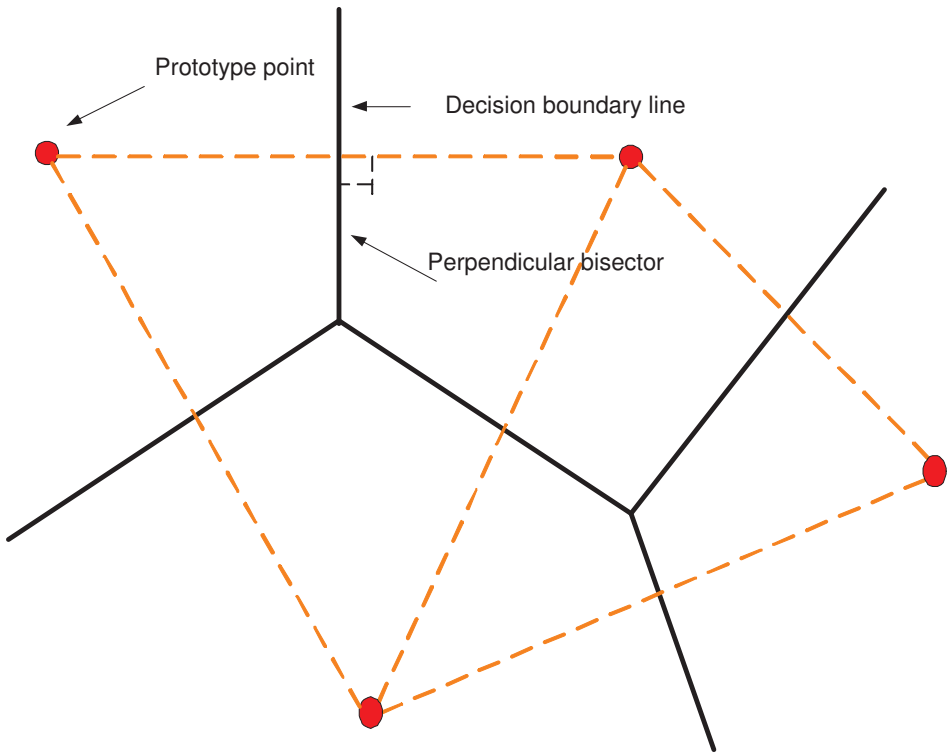


Figure 10: Decision regions of minimum distance classifier

Piecewise linear discriminant functions

- The linear machine has a simple form but suffers the limitation of not being able to separate situations where the decision regions have to be non-convex.
- The examples below show two-class problems where a linear discriminant will fail to separate. They require piece-wise linear discriminant functions.

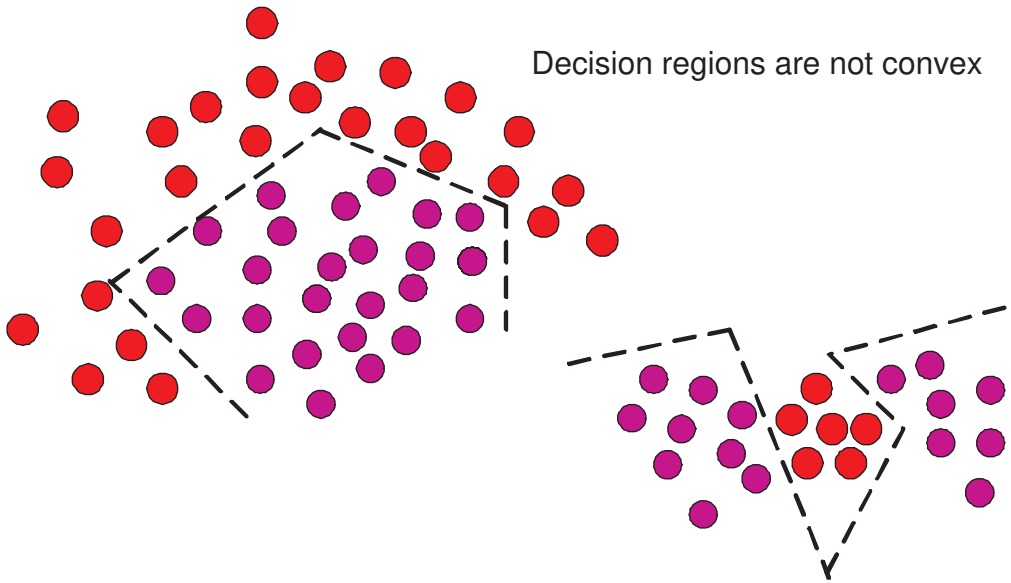


Figure 11: Groups not separable by linear discriminant functions

Piecewise linear discriminant functions

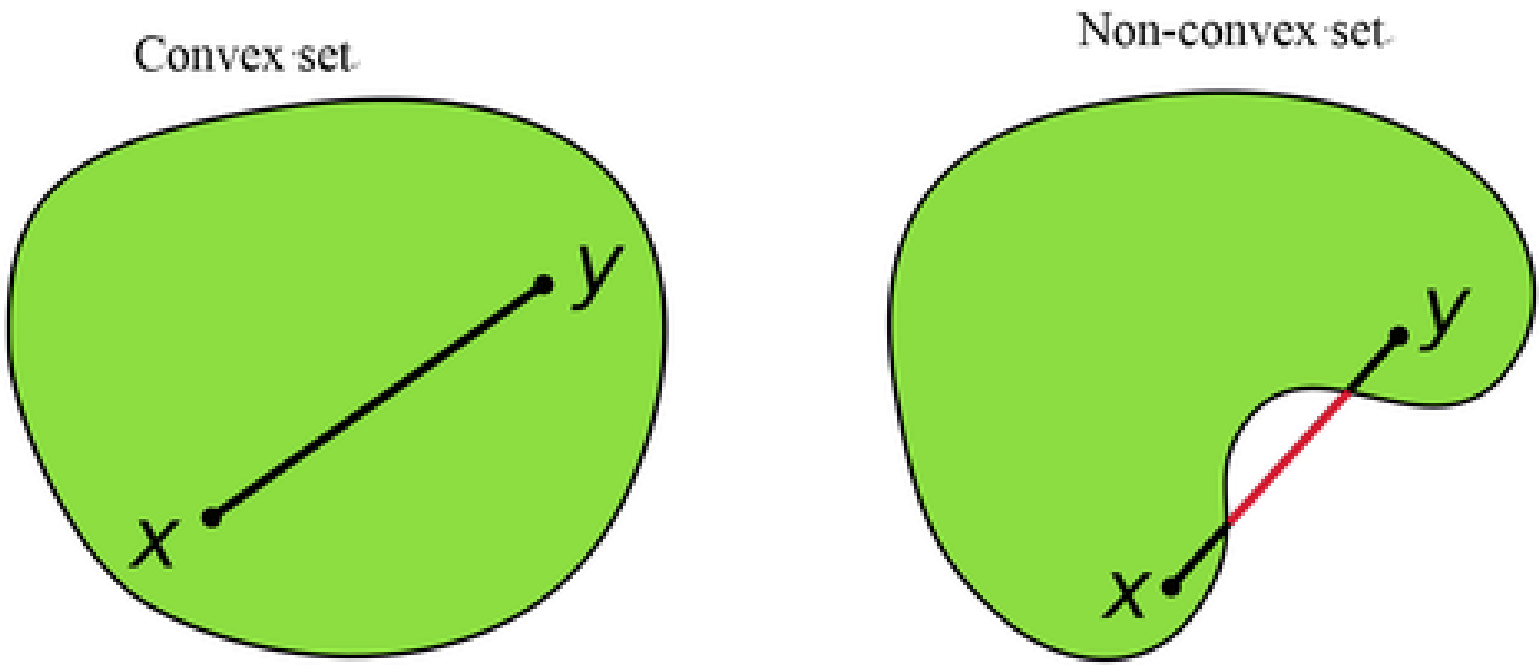


Figure 12: Quick illustration of convex and non-convex regions

Piecewise linear discriminant functions

- We may solve the previous two-class problem by using piece-wise linear discriminant function to generalize the minimum-distance classifier.
- We allow more than one prototype for each class.
- Suppose there are n_i prototypes in class ω_i, $p_i^1, \ldots, p_i^{n_i}$, $i = 1, \ldots, C$.
- The discriminant function which assigns pattern x to class, ω_i, is defined as

$$g_i(x) = \max_{j=1,\ldots,n_i} g_i^j(x)$$

where g_i^j is a linear subsidiary discriminant function, given by

$$g_i^j(x) = x^t p_i^j - \frac{1}{2} p_i^{j\,t}, \quad j = 1, \ldots, n_i;\ i = 1, \ldots, C$$

Kernel methods

Main idea

- Embed given data into a space where the patterns can be discovered as linear relations
- Two steps:
  - Mapping is defined implicitly by a so-called kernel function (depends on domain knowledge regarding pattern in data source)
  - Use general purpose algorithm that is robust
- Algorithm is efficient, requiring computational resource that is polynomial in the size and number of data items; dimension of the embedding space grows exponentially and does not affect computational burden

Kernel methods

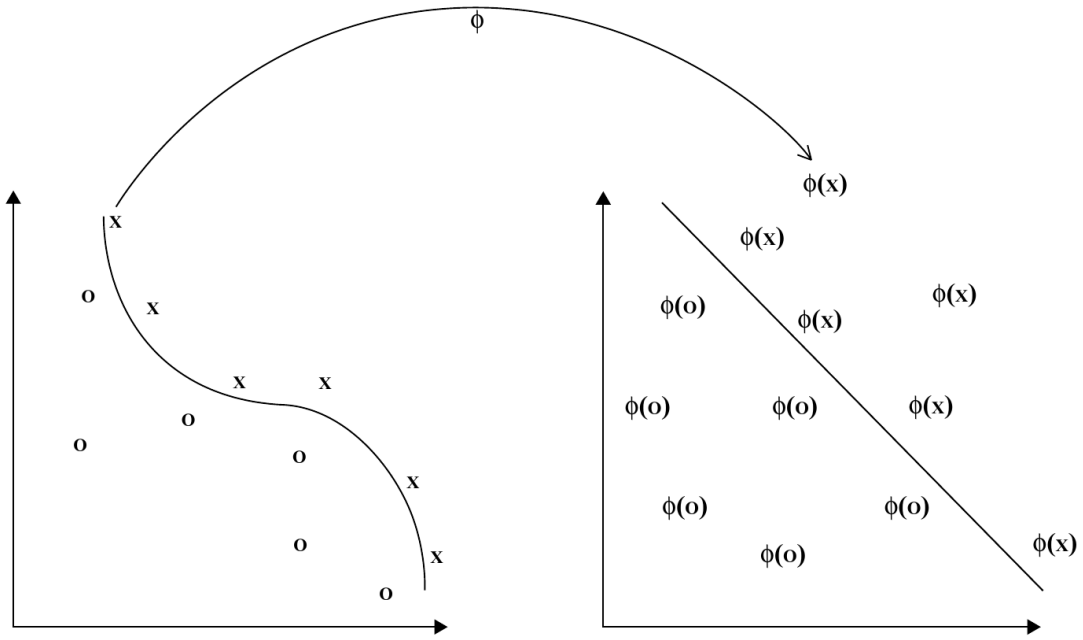


Figure 13: Function ϕ embeds data into a feature space; nonlinear pattern now appears linear

Kernel methods

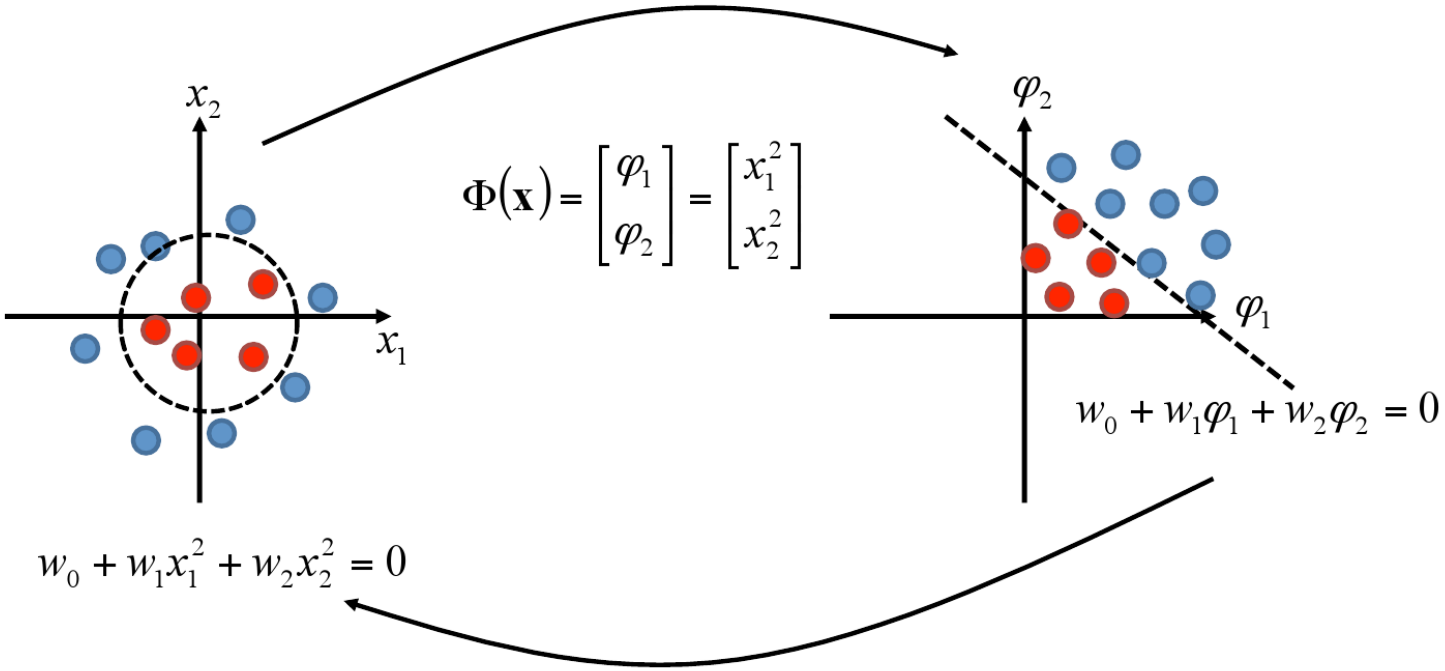


Figure 13: Example of nonlinear mapping in classification problem

Ridge regression revisited

- Formulation is somewhat similar to linear regression;
- This model gives better theoretical guarantees and improved performance partly because we constrain the norm of the weight vector The optimization problem is written compactly as:

$$\min_{\boldsymbol{W}} F(\boldsymbol{W}) = \lambda ||\boldsymbol{W}||^2 + ||\boldsymbol{X}^T\boldsymbol{W} - \boldsymbol{Y}||^2 \tag{3}$$

  where λ is a positive parameter that determines the trade-off between the regularization term $||\boldsymbol{W}||^2$ and the empirical mean squared error; $\boldsymbol{X} \in \mathbb{R}^{N\times m}$ is the matrix of feature vectors, $\boldsymbol{X} = [x_1, \ldots, x_m]$ and $\boldsymbol{W}$ and $\boldsymbol{Y}$ are as defined previously (see lecture slides on regression).

- Optimization problem of Equation is convex and differentiable with a global minimum if and only if

$$\nabla F(\boldsymbol{W}) = 0 \Leftrightarrow (\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I})\boldsymbol{W} = \boldsymbol{X}\boldsymbol{Y} \Leftrightarrow \boldsymbol{W} = (\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{Y} \tag{4}$$

  $\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}$ is always invertible [1]
- The form of the solution weight vector

$$\boldsymbol{W} = (\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{Y}$$

  is the primal solution

[1] because its eigenvalues are sum of non-negative eigenvalues of positive semi-definite matrix $\boldsymbol{X}\boldsymbol{X}^T$ and $\lambda > 0$

Ridge regression revisited

- The solution,

$$\boldsymbol{W} = (\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{Y}$$

can be written as:

$$\boldsymbol{W} = \boldsymbol{X}\alpha \tag{5}$$

where α is

$$(\boldsymbol{G} + \lambda \boldsymbol{I})^{-1}\boldsymbol{Y} \tag{6}$$

and $\boldsymbol{G} = \boldsymbol{X}^T\boldsymbol{X}$ is called the Gram matrix. Each component of $\boldsymbol{G}$ is an inner product, $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$. Thus

$$\boldsymbol{G}_{i,j} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$

- The prediction function is

$$g(x) =< \boldsymbol{W}, x >= \left\langle \sum_{i=1}^{m} \alpha_1 \boldsymbol{x}_i, \boldsymbol{x} \right\rangle = \sum_{i=1}^{m} \alpha_i < \boldsymbol{x}_i, \boldsymbol{x} >= \boldsymbol{Y}^T (G + \lambda \boldsymbol{I})^{-1}\boldsymbol{k}$$

where $k_i =< \boldsymbol{x}_i, \boldsymbol{x} >$

Ridge regression revisited

- We have two forms of solution for the ridge regression (similarly for other regression):

$$\begin{aligned} \boldsymbol{W} &= (\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{Y} : \quad \text{Primal form} \\ \boldsymbol{W} &= \boldsymbol{X}\boldsymbol{\alpha} : \quad \text{Dual form} \end{aligned} \tag{7}$$

  Primal form computes explicitly while the dual expresses solution as linear combination of training samples.

- In primal form we solve an $(N \times N)$ system of equations while in the dual form we solve an $(m \times m)$ system.

- If dimension of features is $N \gg m$ (the number of samples) the computational advantage is obvious

- Key observation: Ridge regression algorithm can be solved in a form that only requires inner products between sample points

Kernel-defined nonlinear mapping

- Consider an embedding map

$$\phi : \boldsymbol{x} \in \mathbb{R}^N \mapsto \phi(\boldsymbol{x}) \in \mathbf{F} \subseteq \mathbb{R}^N \tag{8}$$

- Choose the map ϕ so that it aims to convert the nonlinear relations into linear ones.
- Map ϕ recodes the given dataset, S, into $\{(\phi(\boldsymbol{x}_i), y_i), \ldots (\phi(\boldsymbol{x}_m), y_m)\}$ for the m samples in the dataset.
- Recall that the efficient dual form of the solution to ridge regression entails the Gram matrix which is made up inner products

$$G_{i,j} = \left\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x_j}) \right\rangle \tag{9}$$

- Computational cost of $\boldsymbol{\alpha}$ is $\mathcal{O}(m^3 + m^2 N)$ and that of evaluating the predictor on a new sample is $\mathcal{O}(mN)$
- It turns out that the inner product can be computed directly in the input space rather than first computing $\phi(\boldsymbol{x})$ using a kernel function

Kernel-defined nonlinear mapping

Definition:

A Kernel is a function, κ, that for all $\boldsymbol{x}, \boldsymbol{z} \in S$ satisfies

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle ,$$

where ϕ is a mapping from S to an inner product feature space F

$$\phi : \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) \in \mathbf{F}$$

Kernel-defined nonlinear mapping

Example

Consider a two-dimensional input space $\boldsymbol{x} \subseteq \mathbb{R}^2$ together with the feature map

$$\phi : \boldsymbol{x} = (x_1, x_2) \mapsto \phi(\boldsymbol{x}) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2) \in F = \mathbb{R}^3.$$

Hypothesis of linear functions in F are of the form

$$g(\boldsymbol{x}) = w_{11}x_1^2 + w_{22}x_2^2 + w_{12}\sqrt{2}x_1 x_2$$

$$\begin{aligned}\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle &= \left\langle (x_1^2, x_2^2, \sqrt{2}x_1 x_2), (z_1^2, z_2^2, \sqrt{2}z_1 z_2) \right\rangle \\ &= x_1^2 z_1^2 + x_2^2 x_2^2 + 2x_1 x_2 z_1 z_2 \\ &= (x_1 z_1 + x_2 z_2)^2 = \langle \boldsymbol{x}, \boldsymbol{z} \rangle^2\end{aligned}$$

- Hence $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{x}, \boldsymbol{z} \rangle^2$ is a kernel function with F its corresponding feature space

Kernel-defined nonlinear mapping

Example

Consider a two-dimensional input space $\boldsymbol{x} \subseteq \mathbb{R}^2$ together with the feature map

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

$$\begin{aligned}\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle &= \left\langle (x_1^2, x_2^2, x_1 x_2, x_2 x_1), (z_1^2, z_2^2, z_1 z_2, z_2 z_1) \right\rangle \\ &= x_1^2 z_1^2 + x_2^2 x_2^2 + x_1 x_2 z_1 z_2 + x_2 x_1 z_2 z_1 \\ &= (x_1 z_1 + x_2 z_2)^2 = \langle \boldsymbol{x}, \boldsymbol{z} \rangle^2\end{aligned}$$

- The same kernel computes the inner product of this feature space
- Hence feature space is not uniquely determined by the kernel function

Kernel-defined nonlinear mapping

Table 1: Commonly used kernels ($\gamma, r,$ and d are parameters)

Nonlinearity	Mathematical form: $\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) =$
Linear	$\langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle$
Polynomial	$(\gamma \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle + r)^d; \quad \gamma > 0$
Gaussian (Radial Basis Function - RBF)	$\exp\left(-\gamma \lvert\lvert \boldsymbol{x_i} - \boldsymbol{x_j} \rvert\rvert^2\right); \quad \gamma > 0$
Sigmoid	$\tanh\left(\gamma \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle + r\right)$

Example

Let $\boldsymbol{x} = \begin{bmatrix} 1 & 4 & 6 \end{bmatrix}^T$ and $\boldsymbol{z} = \begin{bmatrix} 3 & 5 & 2 \end{bmatrix}^T$ be two feature vectors that we need to map through the mapping function $\phi(\cdot)$ to some feature space $\mathcal{F}$. Further, let the kernel associated with the feature space be the RBF with parameter $\gamma = 0.2222$. The value of the inner product of the two vectors, $\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle$, in the feature space is easily computed as $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\gamma \lvert\lvert \boldsymbol{x} - \boldsymbol{z} \rvert\rvert^2)$

$$\begin{aligned} \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle &= \kappa(\boldsymbol{x}, \boldsymbol{z}) \\ &= \exp(-0.2222 \times \lvert\lvert \boldsymbol{x} - \boldsymbol{z} \rvert\rvert^2) \\ &= \exp(-4.667) = 9.4 \times 10^{-3} \end{aligned}$$

Try this simple example yourself to ensure you understand the concept

Kernel-defined nonlinear mapping

- Recall Equations (5) and (6) and the comments that followed; each entry of the Gram matrix that appeared in the solution of the ridge regression is an inner product of the data in the input space.
- A mapping, $\phi(\cdot)$, into a high dimensional feature space $\mathcal{F}$, implies that each entry of the Gram matrix can be computed with the appropriate kernel of the feature space

$$\begin{aligned} \boldsymbol{G}_{i,j} &= \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle \\ &= \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned} \tag{10}$$

- Kernalization provides a way of dealing with nonlinear relationships that may exist in the problem (e.g. regression, classification, dimensionality reduction, etc.)

Support vector machine

- Consider a binary classification task with data points $\boldsymbol{x}_i (i = 1, \ldots, m)$ having corresponding labels $y_i = \pm 1$ and decision function

$$g(\boldsymbol{x}) = \text{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) \tag{11}$$

- For a separable dataset all data will be correctly classified if $y_i(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) > 0; \forall i$
- Define canonical hyperplane such that $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 1$ for closest points on one side of separating plane and $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = -1$ for closest points on the other
- Separating plane : $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$ and normal vector is $\frac{\boldsymbol{w}}{||\boldsymbol{w}||}$
- Margin is given by projection of $\boldsymbol{x_1} - \boldsymbol{x_2}$ onto the separating plane
- $\langle \boldsymbol{w}, \boldsymbol{x_1} \rangle + b = 1$ and $\langle \boldsymbol{w}, \boldsymbol{x_2} \rangle + b = -1$; margin is $\gamma = 1/||\boldsymbol{w}||$ (see Fig 14 on next slide)

Support vector machine

Maximize the margin $\gamma = 1/||\boldsymbol{w}||$, by

$$\min \left[\frac{1}{2}||\boldsymbol{w}||^2\right]$$

$$\text{subject to} \quad y_i(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) \geq 1; \forall i.$$

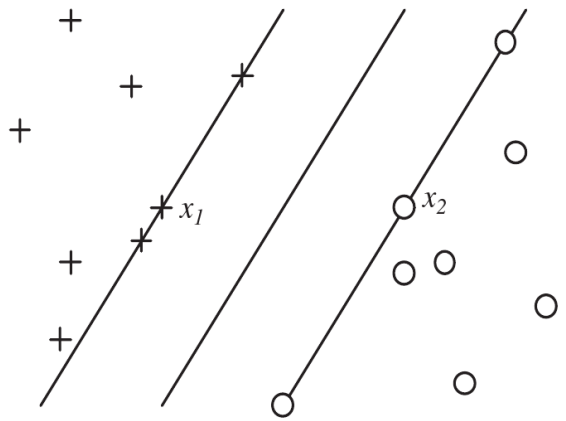


Figure 14: SVM separating hyperplane

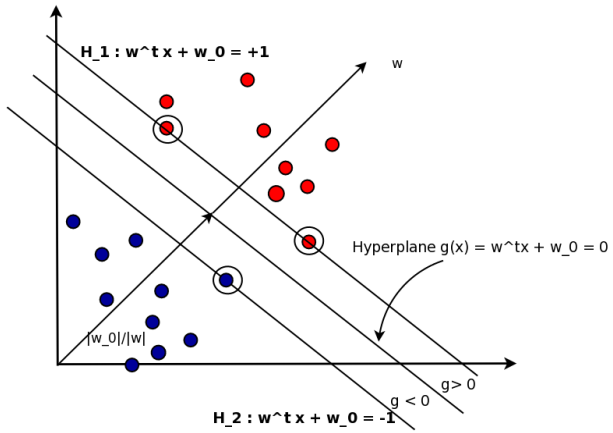


Figure 15: SVM separating hyperplane in detail

Support vector machine

- Learning task reduces to minimizing the primal objective function:

$$\mathcal{L} = \frac{1}{2}(\langle \boldsymbol{w}, \boldsymbol{w} \rangle) - \sum_{i=1}^{m} \alpha_i (y_i (\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b) - 1) \tag{12}$$

  α_i are Lagrange multipliers and $\alpha_i \geq 0$

- After taking derivatives with respect to b and $\boldsymbol{w}$ and appropriate substitution into Equ. 12, we obtain the dual objective function

$$\mathcal{W}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle \tag{13}$$

  to be maximized with respect to the α_i subject to

$$\alpha_i \geq 0 \qquad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{14}$$

- The quadratic program represented by Equ.13 gives optimal separating hyperplane with maximal margin for separable data

Support vector machine

- Equ.13 indicates how we can incorporate the kernel that computes inner product in the feature space after applying a mapping $\phi(\cdot)$; case of inseparable data
- Mapping achieved by

$$\langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle \mapsto \langle \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}) \rangle \tag{15}$$

- Note that the functional form of $\phi(\boldsymbol{x_i})$ need not be known since choice of kernel implicitly defines it:

$$\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) = \langle \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}) \rangle$$

- Kernalized version of Equ. 13 becomes

$$\mathcal{W}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) \tag{16}$$

to be maximized with respect to the α_i subject to

$$\alpha_i \geq 0 \qquad \sum_{i=1}^{m} \alpha_i y_i = 0$$

- For example the kernel could be chosen as $\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp(-\gamma ||\boldsymbol{x_i} - \boldsymbol{x_j}||^2)$
- Decision function for a new test data $\boldsymbol{z}$ is

$$f(\boldsymbol{z}) = sign\left(\sum_{i=1}^{m} y_i \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{z}) + b \right) \tag{17}$$

Support vector machine with soft margin

- Noisy data and outliers can lead to poor generalization
- Reduce the effect by introducing soft margin
- Recall (Equ. 16) that for a given kernel ($\kappa(\cdot,\cdot)$), the learning task is:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \mathcal{W}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) \\ \text{subject to} \quad & \alpha_i \geq 0 \qquad \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned} \tag{18}$$

- Two ways of accounting for noisy data and outlier with
  - Use L_1 norm error and introduce box constrain $0 \leq \alpha_i \leq C$ in Equ (18)
  - Use L_2 norm error and add a small positive constant to the leading diagonal of kernel matrix $\kappa(\boldsymbol{x_i}, \boldsymbol{x_j})$ becomes $\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) + \lambda$ in Equ (18)
- Parameters C and λ are chosen to trade-off between training error and generalization ability; achieve this with validation set; well-known library *libsvm* exposes interface to determine C

Using SVM

Beginners are advised to use the following procedure (see Hsu et al. (2003-2016)):

- Transform data to the format of an SVM package
- Conduct simple scaling on the data
- Consider the RBF kernel $\kappa(\boldsymbol{x_i}, \boldsymbol{x_j} = \exp(-\gamma||\boldsymbol{x_i} - \boldsymbol{x_j}||^2)$ (see Table 42)
- Use cross-validation to find the best parameter C and γ
- Use the best parameter C and γ to train the whole training set
- Test

The guide given by Hsu et al. (2003-2016) is concise and straightforward, and students are strongly encouraged to consult the publication for details.

Bibliography

Alpaydin, E. (2010), *Introduction to Machine Learning*, second edn, The MIT Press, Cambridge Massachusetts.

Ciampiconi, L., Elwood, A., Leonardi, M., Mohamed, A. & Rozza, A. (2023), 'A survey and taxonomy of loss functions in machine learning', online.

Duda, R. O., Hart, P. E. & Stork, D. G. (2001), *Pattern Classification*, Second edn, John Wiley and Sons.

Gönen, M. & Alpaydin, E. (2011), 'Multiple kernel learning algorithms', *Journal of Machine Learning Research* **12**(2011), 2211–2268.

Goodfellow, I., Bengio, Y. & Courville, A. (2016*a*), *Deep Learning*, MIT Press.

Goodfellow, I., Bengio, Y. & Courville, A. (2016*b*), *Deep Learning*, MIT Press.

Hastie, T., Tibshirani, R. & Friedman, J. (2001), *The Elements of Statistical Learning - Data Mining, Inference and Prediction*, Springer Science+Business Media LLC.

Hsu, C.-W., Chang, C.-C. & Lin, C.-J. (2003-2016), Practical guide to support vector classification, Technical report, Department of Computer Science„ National Taiwan University, Taipei 106, Taiwan.

Izenman, A. J. (2008), *Modern Multivariate Statistical Techniques - Regression, Classification and Manifold Learning*, Springer Science+Business Media LLC.

Kellleher, J. D., Namee, B. M. & D'Arcy, A. (2015), *Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples and Case Studies*, The MIT Press, Cambridge Massachusetts.

Mitchell, T. M. (1997), *Machine Learning*, WCB McGraw-Hill.

Mohri, M., Rostamizadeh, A. & Talwalkar, A. (2012), *Foundations of Machine Learning*, MIT Press.

Parnami, A. & Lee, M. (2022), 'Learning from few examples: A summary of approaches to few-shot learning', online.

Webb, A. (2002), *Statistical Pattern Recognition*, Second edn, John Wiley and Sons.

Weinberger, K. Q., Shi, F. & Saul, L. K. (2004), Learning a kernel matrix for nonlinear dimensionality reduction, *in* 'Proceedings, 21st International Conference on Machine Learning', Banff, Canada.