Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 and MoSIG M1 Winter Semester 2016

Lesson 3 10 February 2016

Linear Classifiers and the ROC Curve

Outline

Notation 2

Classification 3

Linear Classifiers as Pattern Detectors 5

ROC Curves 8

The Perceptron Learning Algorithm 10

Definition 10

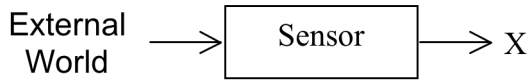
The Algorithm 11

Notation

x_d	A feature. An observed or measured value.
$\vec{X}$	A vector of D features.
D	The number of dimensions for the vector $\vec{X}$
$\vec{y}$	A dependent variable to be estimated.
$\hat{y} = f(\vec{X}, \vec{w})$	A model that predicts $\vec{y}$ from $\vec{X}$
$\vec{w}$	The parameters of the model.
$\{\vec{X}_m\}$ $\{y_m\}$	Training samples for learning.
M	The number of training samples.
K	Number of classes
C_k	The k^{th} class
$\omega_k = \vec{X} \in C_k$	Statement that an observation $\vec{X}$ is a member of class C_k
$\hat{\omega}_k$	An estimation that $\vec{X}$ is a member of class C_k
$\hat{C}_k$	The estimated class
$\tilde{C}_k$	The true class.

Classification

Classification is the process of assigning entities to categories. For machine learning, the "entities" are typically observations of some phenomena. Observations are produced by some form of sensor.



The result is a feature vector, $\vec{X}$, of D properties or features $\vec{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{pmatrix}$

Features can be numerical (reals, floats, integers, etc) or symbolic labels.
Numerical features are drawn from an infinite set of values and obey an ordering relation, ">".

Symbolic features may be from a finite set or an infinite set and do not necessarily obey an ordering relation. There can be other relations between symbolic features.

Our problem is to build a box that maps the observation, $\vec{X}$ into an estimate $\hat{C}_k$ of the class C_k from a set of K possible classes.

Where $\hat{\omega}_k$ is a statement that asserts that $\vec{X}$ is a member of class $\hat{C}_k$, one of the K possible classes. $\hat{C}_k \in \{C_k\}$

Almost all current classification techniques require the set $\{C_k\}$ to be predefined.
An open research problem is how to design classification algorithms that allow $\{C_k\}$ to be an open set that grows with experience.

Let $\tilde{C}_k$ be the True class of $\vec{X}$. Then the classification is TRUE or T if $\hat{C}_k = \tilde{C}_k$ and the classification is FALSE or F if $\hat{C}_k \neq \tilde{C}_k$.
In other words, the classification is TRUE or T if $\hat{\omega}_k = \omega_k$ and FALSE or F if $\hat{\omega}_k \neq \omega_k$.

if $\hat{\omega}_k = \omega_k$ then T else F.

When $\hat{\omega}_k \neq \omega_k$ the classification is an Error.
Our task is to minimize the number of Errors.
We can minimize the number of errors by explicitly maximizing the probability $P(\vec{X} \in C_k \mid \vec{X})$

$$\hat{C}_k = \arg\max_{C_k} \left\{ P(\vec{X} \in C_k \mid \vec{X}) \right\}$$

because the probability of TRUE plus the probability of FALSE is 1.

$$P(\vec{X} \in C_k \mid \vec{X}) + P(\vec{X} \notin C_k \mid \vec{X}) = 1$$

If we explicitly calculate the probability, this is called a "generative" method.
If we do not explicitly calculate the probability, the method is "discriminative".

In some problems, there is a notion of "cost" for errors that is different than the cost of true classifications.
In this case we will seek to minimize the cost of an error rather than the number of errors by biasing the classification.

The classification function can be decomposed into two parts: $\hat{C}_k \leftarrow d\left(\vec{g}\left(\vec{X}\right)\right)$
where $\vec{g}\left(\vec{X}\right)$ is a discrminant and $d(-)$ is a decision function.

$\vec{g}\left(\vec{X}\right)$: A discriminant function that transforms $\vec{X} \rightarrow R^K$
$d(-)$: A decision function $R^K \rightarrow \hat{C}_k$

The discriminant is typically a vector of functions:

$$\vec{g}(\vec{X}) = \begin{pmatrix} g_1(\vec{X}) \\ g_2(\vec{X}) \\ \vdots \\ g_K(\vec{X}) \end{pmatrix}$$

Linear Classifiers as Pattern Detectors

Linear classifiers are widely used to define pattern "detectors". These are used in computer vision, for example to detect faces, road signs, publicity logos, or other patterns of interest.

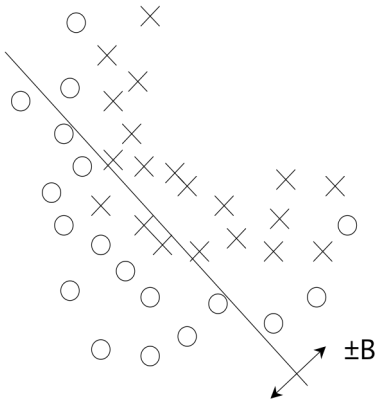
A pattern detector can be seen as a classifier with K=2.

Class k=1: The target pattern.
Class k=2: Everything else.

The pattern detector is learned as a detection function $g(\vec{X})$ followed by a decision rule, d(). The detection function is learned from a set of training data composed of N sample observations $\{\vec{X}_m\}$ where each sample observation is labeled with an indicator variable $\{y_m\}$

$y_m = +1$ for examples of the target pattern (class 1)
$y_m = -1$ for all other examples (class 2)

As we have seen, a linear detection function is a hyperplane in a D dimensional feature space, $\vec{X}$ that provides a decision boundary between the target class and everything else.



A hyperplane is a set of points such that $w_1x_1 + w_2x_2 + ... + w_Dx_D + b = 0$

Note that $\vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{pmatrix}$ is the normal to the hyperplane.

When $\vec{w}$ is normalized to unit length, $\|\vec{w}\| = 1$, then

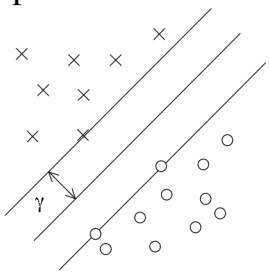
$b = -\vec{w}^T \vec{X}$ is the perpendicular distance to the origin for a hyperplane that includes the point $\vec{X}$

if $\|\vec{w}\| \neq 1$ then we can normalize with $\|\vec{w}\| \leftarrow \frac{\vec{w}}{\|\vec{w}\|}$ and $b \leftarrow \frac{b}{\|\vec{w}\|}$

We will typically learn the decision surface from a set of *M* training data $\{\vec{X}_m\}$, $\{y_m\}$.

When the training data are perfectly separated the data is said to be "separable". Otherwise, the data is said to be non-separable.

The separation is defined using the margin: , γ.
The "margin", γ, is the smallest separation between the two classes.



If we have normalized the coefficients of the hyperplane such that $\|\vec{w}\| = 1$
then the distance of any sample point $\vec{X}_m$ from the hyper-plane, $\vec{w}$.

$$d_m = y_m \cdot \left(\vec{w}^T \vec{X}_m + w_0\right)$$

The margin is the minimum distance of a training sample to the decision surface.

$$\gamma = \min\{y_m \cdot \left(\vec{w}^T \vec{X}_m + w_0\right)\}$$

As in previous lesson, we can simplify the notation by including the constant term in our vectors.

$$\vec{X} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix} \text{ and } \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix} \text{ where } w_0 \text{ represents b.}$$

The discriminant can be written: $\vec{w}^T \vec{X} = 0$

Observations for which $\vec{w}^T \vec{X} > 0$ are assumed to be members of the target class.
This will be called POSITIVE or P.

Otherwise, $\vec{w}^T\vec{X} \le 0$ and the detection is said to be NEGATIVE or N.

This gives a decision rule: if $\vec{w}^T\vec{X} > 0$ then P else N

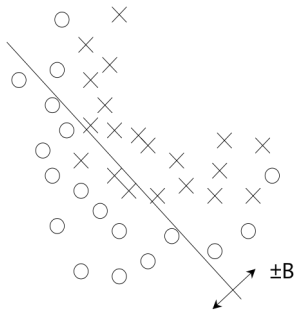
As we saw before, the Classification can be TRUE or FALSE.

if $\hat{C}_k = \tilde{C}_k$ then T else F

This gives: if ($\tilde{C}_1$ and P) or ($\tilde{C}_2$ and F) then TRUE else FALSE.

($\hat{C}_1$ and $\tilde{C}_1$) is a TRUE POSITIVE or TP
($\hat{C}_2$ and $\tilde{C}_2$) is a TRUE NEGATIVE or TN
($\hat{C}_1$ and $\tilde{C}_2$) is a FALSE POSITIVE or FP
($\hat{C}_2$ and $\tilde{C}_1$) is a FALSE NEGATIVE or FN

We can add an additional bias term, B, that can act as an adjustable gain that sets the sensitivity of the detector. The bias term allows us to trade False positives for False negatives.



$g(\vec{X},\vec{w}) = \vec{w}^T\vec{X} + B$

A variety of techniques exist to estimate the plane.
Popular techniques include

1) Least Squares estimation
2) Regression
3) Perceptrons
4) Bayesian Discriminant functions
5) Fisher Discriminant analysis.

The best choice can depend on the nature of the data as well as the way in which the linear detection function is used. In many cases a fast sub-optimal technique may be preferred to a more expensive optimal technique.
To better understand we need a tool to explore the trade-off between making false detections (false positives) and missed detections (false negatives). The Receiver Operating Characteristic (ROC) provides such a tool

ROC Curves

Two-class linear classifiers have long been used for signal detection problems in communications. In the case of radio communications, the noise is typically additive, Gaussian and independent of the signal, and the Bayesian Classifier reduces to a linear classifier.

Historically two class linear classifiers have been used to demonstrate optimality for some signal detection methods. The quality metric that is used is the Receiver Operating Characteristic (ROC) curve. This curve can be used to describe or compare any method for signal or pattern detection.

The ROC curve is generated by adding a variable Bias term to a linear discriminant

$$g(\vec{X},\vec{w}) = \vec{w}^T\vec{X} + B$$

The bias term, B, is swept through a range of values.
Changing B changes the ratio of true positive detection to false positives.

The resulting curve is called a Receiver Operating Characteristics (ROC) curve.

The ROC plots True Positive Rate (TPR) against False Positive Rate (FNR) as a function of B for the training data $\{\vec{X}_m\}$, $\{y_m\}$.

For each training sample, the detection as either Positive (P) or Negative (N)

IF $\vec{w}^T\vec{X}_m + B > 0$ THEN P else N

The detection can be TRUE (T) or FALSE (F) depending on the indicator y_m

IF $y_m \cdot (\vec{w}^T\vec{X}_m + B) > 0$ THEN T else F

Combining these two values, any detection can be a True Positive (TP), False Positive (FP), True Negative (TN) or False Negative (FN).

For the M samples of the training data $\{\vec{X}_m\}$, $\{y_m\}$ let us define:
#P as the number of Positives,
#N as the number of Negatives,
#T as the number of True and
#F as the number of False,

From this we can define:

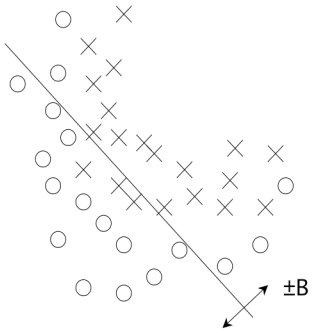
#TP as the number of True Positives,
#FP as the number of False Positives,
#TN as the number of True Negative,
#FN as the number of False Negatives.

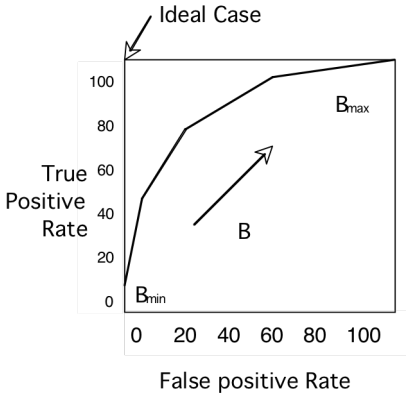
Note that #P = #TP + #FN
And #N = #FP+ #TN

The True Positive Rate (TPR) is $TPR = \frac{\#TP}{\#P} = \frac{\#TP}{\#TP+\#FN}$

The False Positive Rate (FPR) is $FPR = \frac{\#FP}{\#N} = \frac{\#FP}{\#FP+\#TN}$

The ROC plots the TPR against the FPR as a bias B is swept through a range of values.





When B is large, all the samples are detected as N, and both the TPR and FPR are 0. As B decreases both the TPR and FPR increase. Normally TPR is larger than FPR for any B. If TPR and FPR are equal, then the detector is no better than chance.

The more the curve approaches the upper left corner the better the detector.
The ROC is a powerful descriptor for the "goodness" of a linear classifier.
For a target class C_1

- a Positive (P) detection is the decision that $E \in C_1$
- a Negative (N) detection is the decision that $E \in C_2$

		$y_m \cdot (\vec{W}^T \vec{X}_m + B) > 0$	
		T	F
$\vec{W}^T \vec{X}_m + B > 0$	P	True Positive (TP)	False Positive (FP)
	N	False Negative (FN)	True Negative (TN)

The Perceptron Learning Algorithm

History:
The perceptron is an incremental learning algorithm for linear classifiers invented by Frank Rosenblatt in 1956. The perceptron is an on-line learning method in which a linear classifier is improved by its own errors.

Definition
A perceptron learns a set of possible hyper-planes to separate training samples.
When are the training samples are separable, the algorithm uses the errors to update a plane until there are no more errors. When the training data is non-separable, the method may not converge, and must be stopped after a certain number of iterations.

The learning can be incremental. New training samples can be used to update the perceptron.

Assume a training set of M observations $\{\vec{X}_m\}$ $\{y_m\}$ where

$$\vec{X}_m = \begin{pmatrix} x_{1m} \\ x_{2m} \\ \vdots \\ x_{Dm} \end{pmatrix} \text{ and } \vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{pmatrix}$$

The indicator variable, $\{y_m\}$, for each sample,

$y_m = +1$ for examples of the target pattern (class 1)
$y_m = -1$ for all other examples (class 2)

Note that for all correctly classified training samples:

$$y_m \cdot \left(\vec{w}^T \vec{X}_m + w_0\right) > 0$$

The algorithm requires a learning rate, α.

The Algorithm

Algorithm:

$\vec{w}^{(0)} \leftarrow 0;\ w_0 \leftarrow 0,\ i \leftarrow 0;$
R $\leftarrow$ max { $\|\vec{X}_m\|$}
WHILE update DO
  update $\leftarrow$ FALSE;
  FOR *m = 1* TO M DO
    IF $y_m \cdot \left(\vec{w}^{(i)T}\vec{X}_m + w_0^{(i)}\right) \le 0$ THEN
      update $\leftarrow$ TRUE
      $\vec{w}^{(i+1)} \leftarrow \vec{w}^{(i)} + \alpha \cdot y_m \cdot \vec{X}_m$
      $w_0^{(i+1)} \leftarrow w_0^{(i)} + \alpha \cdot y_m \cdot R^2$
      $i \leftarrow i + 1$
    END IF
  END FOR
END WHILE.

The decision rule for any observation, $\vec{X}$, then becomes:

if $\text{sgn}\left(\vec{w}^{(i)T}\vec{X} + w_0^{(i)}\right) > 0$ then POSITIVE else N.

where

$$\text{sgn}(z) = \begin{cases} 1 & \text{if } z \ge 0 \\ -1 & \text{if } z < 0 \end{cases}$$

After each stage the margin, γ_m, for each sample, m, is

$$\gamma_m = y_m \cdot \left(\vec{w}^{(i)T}\vec{X}_m + w_0^{(i)}\right)$$

The quality of a perceptron is given by the histogram of its margins, h(γ_m)
If the data is not separable, then the Perceptron will not converge, and continue to loop. Thus it is necessary to have a limit the number of iterations.