Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 / MoSIG M1 Second Semester 2014/2015

Lesson 17 10 April 2015

Boosted Learning

Contents

Bibliographical Sources :
"Pattern Recognition and Scene Analysis", R. E. Duda and P. E. Hart, Wiley, 1973.
"Pattern Recognition and Machine Learning", C. M. Bishop, Springer Verlag, 2006.

Notation

x	a variable
X	a random variable (unpredictable value)
V	The number of possible values for X (Can be infinite).
$\vec{x}$	A vector of D variables.
$\vec{X}$	A vector of D random variables.
D	The number of dimensions for the vector $\vec{x}$ or $\vec{X}$
E	An observation. An event.
C_k	The class k
k	Class index
K	Total number of classes
ω_k	The statement (assertion) that $E \in C_k$

$P(\omega_k) = P(E \in C_k)$ Probability that the observation E is a member of the class k.

$P(X)$	Probability density function for X
$p(\vec{X})$	Probability density function for $\vec{X}$
$p(\vec{X} \mid \omega_k)$	Probability density for $\vec{X}$ the class k. $\omega_k = E \in C_k$.
$\{\vec{X}_n\}$ $\{y_n\}$	N Training samples labeled with an indicator variable.

For two class problems, the indicator is: $y_n = +1$ for target class and $y_n = -1$ for other

A Committee of Linear Detection Functions

One of the more original ideas in machine learning the last decade is the discovery of a method by to learn a committee of classifiers by boosting. A boosted committee of classifiers can be made arbitrarily good: Adding a new classifier always improves performance.

Assume training data composed of N sample observations $\{\vec{X}_n\}$ and indicator variables, $\{y_n\}$ where

$y_n = +1$ for examples of the target pattern (class 1)

$y_n = -1$ for all other examples.

and each sample, $\vec{X}_n$ is a D dimensional vector,

Our goal is to learn a set of linear detection functions that separates observations of the target class from everything else.

A linear detection function is hyperplane is a set of points such that

$$w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + d = 0$$

For convenience, we will use homogeneous coordinates so that:

$$\vec{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{pmatrix} \text{ and } \vec{W} = \begin{pmatrix} w_1 \\ \vdots \\ w_D \\ d \end{pmatrix}$$

We will learn a committee of M detection function $g_m(\vec{X}) = \vec{W}_m^T \vec{X}$

Each detection function will vote for the class of an observation:

We can express this using the function sgn() :

The "vote" for the classifiers is $\text{sgn}(\vec{W}^T \vec{X}) = \begin{cases} 1 & \text{if } \vec{W}^T \vec{X} \geq 0 \\ -1 & \text{if } \vec{W}^T \vec{X} < 0 \end{cases}$.

We can sum the votes for M detection function with $\sum_{m=1}^{M} \text{sgn}(\vec{W}_m^T \vec{X})$

Thus, for a committee of M detection functions, the decision rule is:

if $\sum_{m=1}^{M} \text{sgn}(\vec{W}_M^T \vec{X}) > 0$ Class 1 (True) else class 2 (False).

We can bias the classifier to prefer class 1 or class 2 by adding a bias, B to each detection function.

For a biased committee of M classifiers, the decision rule is:

if $\sum_{M=1}^{M} \text{sgn}(\vec{W}_m^T \vec{X} + B) > 0$ Class 1 (True) else class 2 (False).

Learning a committee of detection functions with Boosting

We can learn a committee of classifiers using boosting. Boosted learning allows us to improve a committee by adding new classifiers

For this we will add weight to the indicators variables $\{y_n\}$ for samples that are misclassifed. Initially, at i=0, all the weights in $\{y_n^0\}$ are 1 or -1.

We can use a variety of different methods to estimate the linear detection functions, however it is common to use least squares:
In this case, the N training samples to compose a matrix **X**.

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \cdots & \cdots & \cdots & \vdots \\ x_{D1} & x_{D2} & \ddots & x_{DN} \\ 1 & 1 & \cdots & 1 \end{pmatrix} \text{ (D+1 rows by N columns)}$$

For each iteration, the indicator variables give a vector

$$Y^i = \begin{pmatrix} y_1^i \\ y_2^i \\ \vdots \\ y_N^i \end{pmatrix} \text{ (vector of N indicator variables).}$$

The weight of each indicator variable will be raised for misclassified samples by adding $\text{sgn}(y_n)$.

For each cycle, *i*, we learn the i^{th} the detection function $\vec{W}_i$ from

$$\vec{W}_i = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\vec{Y}^i$$

After each new detector is learned, we re-weight the indicator variables by testing the committee for each training sample and giving more weight to improperly classified training samples.

FOR n = 1 TO N: IF $(y_n^i \cdot \sum_{m=1}^{i} \text{sgn}(\vec{W}_m^T \vec{X}_n)) < 0$ THEN $y_n^{i+1} \leftarrow y_n^i + \text{sgn}(y_n^i)$

By adding sgn(y), negative variables are made more negative, positive more positive. We then learn the next classifier using the re-weighted indicator variables.

$$\vec{W}_{i+1} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\vec{Y}^{i+1}$$

We note that a detection is True if

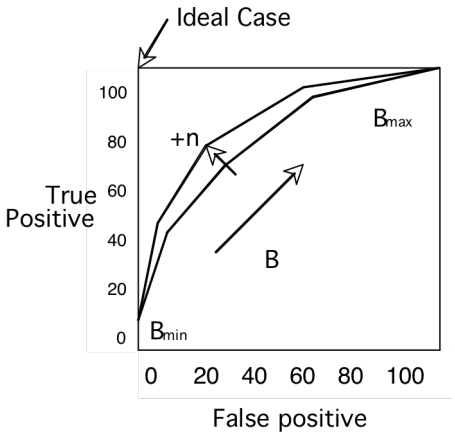
IF $y_n \cdot \sum_{m=1}^{i} \text{sgn}(\vec{W}_m^T \vec{X}_n) > 0$ THEN True ELSE False.

We can count the number of TRUE and FALSE detections as

For n=1 to N: IF $y_n \cdot \sum_{m=1}^{i} \text{sgn}(\vec{W}_m^T \vec{X}_n) > 0$ THEN T ←T+1 ELSE F←F+1.

ROC Curve

As we saw Wednesday, the ROC plots the True Positive Rate (TPR) against False Positive Rate (FPR) for a classifier as a function of the global bias B.



The probability of error for a committee detectors can be computed as

$$P(Error) = \frac{F}{N} = \frac{\#FP + \#FN}{N}$$

Where N is the number of training samples, and F is the number of False detections within the N training samples.

The Boosting theorem states that adding a new boosted detectors to a committee always improves the committee's ROC curve. We can continue adding classifiers until we obtain a desired rate of false positives and false negatives.

However, in general, the improvement provided for each new classifier becomes progressively smaller. We can end up with a very very large number of classifiers.

Treating each feature as a detection functions.

In some cases it is convenient to use the individual features (properties), x_d of $\vec{X}$ as detection functions. In this case our linear decision surface is simply a Kronecker delta:

$$\vec{W} = \vec{\delta}_d = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

Thus we can replace least squares estimation with a simple selection process for the "best" feature x_d as selected by $\vec{W} = \vec{\delta}_d$.

For each iteration, i, the new detection function W_i, is chosen as the feature, d, that gives the most correct classifications for the boosted data.

$$\vec{W}_i \leftarrow \vec{\delta}_d = \arg\max_{\vec{\delta}_d} \left\{ \sum_{n=1}^{N} (y_n^i \cdot \vec{\delta}_d^T X_n) \right\}$$

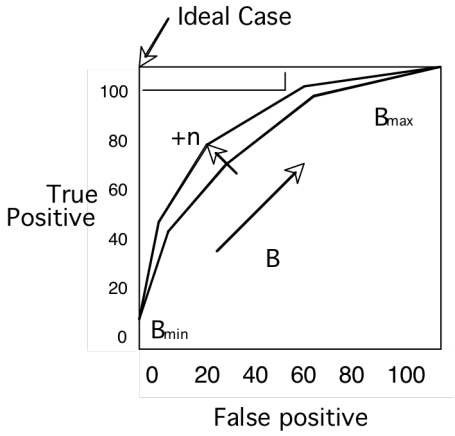
This feature can be removed from the set of candidate features $\{\vec{\delta}_d\}$ after selection.

This is the technique that is used in the Viola-Jones Face detector.

Learning a multi-stage cascade of classifiers

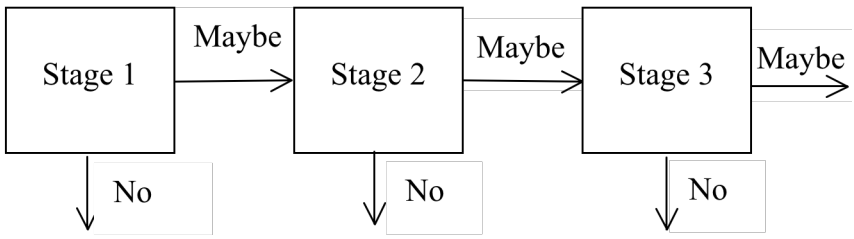
We can optimize the computation time by separating the committee into a multi-stage cascade of committees

We can use a bias B to construct a committee that is biased to have avoid missing true positives (high True Positive Rate) at the cost of accepting many False Positives (high False Positive Rate).



Thus each stage acts as a "filter" to remove true negatives.

We can the construct each stage to eliminate the false positives that passed the previous stages. Later stages are more expensive but are used less often.



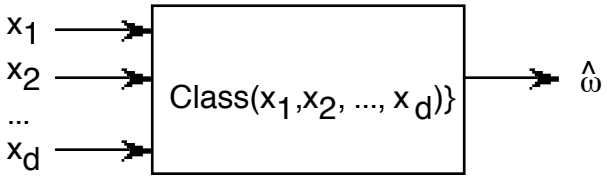
For each stage, S, we set a target error rate : (FPR_s, FNR_s).

We then train the committee using all positive samples from that passed stages from 1 to S-1.

Each stage acts as a filter, rejecting a large number of easy negative cases, and passing the hard cases to the next stage. The stages become progressively more expensive, but are used progressively less often. Globally the computation cost decreases dramatically.

Bayesian Linear Discriminant Functions

As we saw in earlier lessons, a Bayesian classifier maps a set of features $\vec{X}$ from an Observation, E into a class C_k from a set of K possible Classes.



Let ω_k be the proposition that the event belongs to class k: $\omega_k = E \in C_k$

ω_k Proposition that event E $\in$ the class k

In order to minimize the number of mistakes, we will maximize the probability that $\omega_k \equiv E \in T_k$

$$\hat{\omega}_k = \arg\max_{\omega_k} \left\{ P(\omega_k \mid \vec{X}) \right\}$$

We will call on two tools for this:

1) Baye's Rule : $P(\omega_k \mid \vec{X}) = \frac{p(\vec{X} \mid \omega_k)}{p(\vec{X})} P(\omega_k)$

2) Normal Density Functions $p(\vec{X}) = \mathcal{N}(\vec{X}; \vec{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}} \det(\Sigma)^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{X}-\vec{\mu})^T \Sigma^{-1} (\vec{X}-\vec{\mu})}$

The classification function can be decomposed into two parts: d() and $g_k()$:

$$\hat{\omega}_k = d\left(g_k\left(\vec{X}\right)\right)$$

$g_k\left(\vec{X}\right)$: A discriminant function : $R^D \rightarrow R^K$

d() : a decision function $R^K \rightarrow \omega_k$

The discriminant is a vector of functions: $\vec{g}(\vec{X}) = \begin{pmatrix} g_1(\vec{X}) \\ g_2(\vec{X}) \\ \vdots \\ g_K(\vec{X}) \end{pmatrix}$

Quadratic discrimination functions can be derived directly from $p(\omega_k \mid \vec{X})$

$$p(\omega_k \mid \vec{X}) = \frac{P(\vec{X} \mid \omega_k)p(\omega_k)}{P(\vec{X})}$$

To minimize the number of errors, we will choose k such that

$$\hat{\omega}_k = \underset{\omega_k}{\arg\max}\{\frac{P(\vec{X} \mid \omega_k)p(\omega_k)}{P(\vec{X})}\}$$

but because P(X) is constant for all k, it is common to use a likelihood function:

$$\hat{\omega}_k = \underset{\omega_k}{\arg\max}\{P(\vec{X} \mid \omega_k)p(\omega_k)\}$$

This is called a "Maximum Likelihood" classifier.
The functions $g_k()$ are commonly constructed from the Log of the likelihood:

$$g_k(X) = Log\{P(\vec{X} \mid \omega_k)P(\omega_k)\}$$

as Log is a monotonic function.

For a Gaussian (Normal) density function

$$p(\vec{X} \mid \omega_k) = \mathcal{N}(\vec{X}; \vec{\mu}_k, \Sigma_k)$$

$$Log(p(\vec{X} \mid \omega_k)) = Log(\frac{1}{(2\pi)^{\frac{D}{2}} \det(\Sigma_k)^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{X}-\vec{\mu}_k)^T \Sigma_k^{-1}(\vec{X}-\vec{\mu}_k)})$$

$$Log(p(\vec{X} \mid \omega_k)) = -\frac{D}{2}Log(2\pi) - \frac{1}{2}Log\{Det(\Sigma_k)\} - \frac{1}{2}(\vec{X}-\vec{\mu}_k)^T \Sigma_k^{-1}(\vec{X}-\vec{\mu}_k)$$

We can observe that $-\frac{D}{2}Log(2\pi)$ can be ignored because it is constant for all k.

The discrimination function becomes:

$$g_k(\vec{X}) = -\frac{1}{2}Log\{\det(\Sigma_k)\} - \frac{1}{2}(\vec{X}-\vec{\mu}_k)^T \Sigma_k^{-1}(\vec{X}-\vec{\mu}_k) + Log\{p(\omega_k)\}$$

Different families of Bayesian classifiers can be defined by variations of this formula.

This becomes more evident if we reduce the equation to the standard (canonical) form for a quadic polynomial.

The term $(\vec{X}-\vec{\mu}_k)^T\Sigma_k^{-1}(\vec{X}-\vec{\mu}_k)$ can be rewritten as :

$$\vec{X}^T\Sigma_k^{-1}\vec{X}+-2(\Sigma_k^{-1}\vec{\mu}_k)^T\vec{X}++\vec{\mu}_k{}^T\Sigma_k^{-1}\vec{\mu}_k$$

Demonstration:

$$(\vec{X}-\vec{\mu}_k)^T\Sigma_k^{-1}(\vec{X}-\vec{\mu}_k) = \vec{X}^T\Sigma_k^{-1}\vec{X}-\vec{X}^T\Sigma_k^{-1}\vec{\mu}_k-\vec{\mu}_k{}^T\Sigma_k^{-1}\vec{X}+\vec{\mu}_k{}^T\Sigma_k^{-1}\vec{\mu}_k$$

We note that $\vec{X}^T\Sigma_k^{-1}\vec{\mu}_k=\vec{\mu}_k{}^T\Sigma_k^{-1}\vec{X}$

and thus : $-\vec{X}^T\Sigma_k^{-1}\vec{\mu}_k-\vec{\mu}_k{}^T\Sigma_k^{-1}\vec{X}= -(2\Sigma_k^{-1}\vec{\mu}_k)^T\vec{X}$

we define: $\vec{W}_k=-2\Sigma_k^{-1}\vec{\mu}_k$

to obtain $-\vec{X}^T\Sigma_k^{-1}\vec{\mu}_k-\vec{\mu}_k{}^T\Sigma_k^{-1}\vec{X}= \vec{W}_k{}^T\vec{X}$

Let us also define $D_k=-\frac{1}{2}\Sigma_k^{-1}$

The remaining terms are constant. Let us defined the constant

$$d_k = -\frac{1}{2}\vec{\mu}_k{}^T\Sigma_k^{-1}\vec{\mu}_k-Log\{\det(\Sigma_k)\}+Log\{p(\omega_k)\}$$

which gives a quadratic polynomial

$$g_k(\vec{X})=\vec{X}^TD_k\vec{X}+\vec{W}_k^T\vec{X}+d_k$$

where: $D_k=-\frac{1}{2}C_k^{-1}$

$\vec{W}_k=-2\Sigma_k^{-1}\vec{\mu}_k$

and $d_k = -\frac{1}{2}\vec{\mu}_k{}^T\Sigma_k^{-1}\vec{\mu}_k-Log\{\det(\Sigma_k)\}+Log\{p(\omega_k)\}$

A set of K discrimination functions $g_k(\vec{X})$ partitions the space $\vec{X}$ into a disjoint set of regions with quadratic boundaries.

The boundaries between classes are defined by points for which

$$g_i(\vec{X}) = g_j(\vec{X}) \geq g_k(\vec{X}) \; \forall k \neq i, j$$

Thus the detection function between class i and j can be: $g_{ij}(\vec{X}) = g_i(\vec{X}) - g_j(\vec{X})$
with the decision rule :

IF $g_{ij}(\vec{X}) > 0$ THEN C_i else C_j

This boundary is a 2nd order (quadric) polynomial in D dimensions.

Under certain conditions, the quadratic discrimination function can be simplified by eliminating either the quadratic or the linear term.

For example if $\vec{N}_s >> \vec{N}_k$ then the term Σ_k will be nearly constant for all k.
In this case, the discrimination function can be reduced to a linear equation.

$$g_k(\vec{X}) = \vec{W}_k^T \vec{X} + d_k$$

Bayesian Linear Detectors:

Suppose that we have two classes with mean and covariance ($\vec{\mu}_1$, Σ_1), and ($\vec{\mu}_2$, Σ_2). These can be used to define two linear discriminant functions:

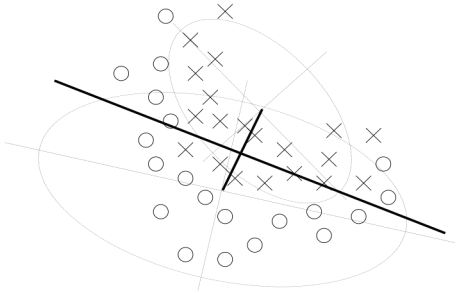
Let $g_1(\vec{X}) = \vec{W}_1^T \vec{X} + d_1$ and $g_2(\vec{X}) = \vec{W}_2^T \vec{X} + d_2$

where : $\vec{W}_k = \Sigma_k^{-1} \vec{\mu}_k$

and $d_k = -\frac{1}{2}(\mu_k^T \Sigma_k^{-1} \mu_k) - \frac{1}{2} Log\{\det(\Sigma_k)\} + Log\{p(\omega_k)\}$

The decision boundary is

$$g_1(\vec{X}) - g_2(\vec{X}) = 0$$
$$(\vec{W}_1^T - \vec{W}_2^T)\vec{X} + d_1 - d_2 = 0$$
$$(\Sigma_1^{-1}\vec{\mu}_1 - \Sigma_2^{-1}\vec{\mu}_2)^T X + d_1 - d_2 = 0$$



The direction is determined by the vector between the center of gravities of the two classes, weighted by the inverse of the covariance matrices.

This approach is based on the assumption that the two classes are well modeled by Normal density functions. This assumption is not reasonable in many cases.
If one of the classes is not well modeled as a normal, the results can be unreliable.

In some other cases, the data are so well separated that a large variety of hyperplanes can be used. In this case it can be interesting to use a simpler learning method.