Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 / MoSIG M1 Second Semester 2013/2014

Lesson 19 25 April 2014

Linear Detectors and Boosted Learning

Contents

Linear Classifiers as Pattern Detectors 2

Least squares estimation of a hyperplane 3

A Committee of Boosted Classifiers..................................5

Learning a Committee of Classifiers with Boosting......................6

ROC Curve..8

Learning a Multi-Stage Cascade of Classifiers 9

Sources Bibliographiques :
"Neural Networks for Pattern Recognition", C. M. Bishop, Oxford Univ. Press, 1995.

Linear Classifiers as Pattern Detectors

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

In the case of pattern detectors, K=2.

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

The detector is learned form a set of training data training composed of M sample observations $\{\vec{X}_m\}$ where each sample observation is labeled with an indicator variable

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

Our goal is to build a hyper-plane that provides a best separation of class 1 from class 2. The hyper plane has the form:

$$\vec{W}^T\vec{X} + B = 0$$

A hyperplane is a set of points such that

$$w_1x_1 + w_2x_2 + ... + w_Dx_D + B = 0$$

The decision rule is IF $\vec{W}^T\vec{X} + B > 0$ THEN $E \in C_1$ else $E \notin C_1$

For convenience we can add a "0th" term to X and W, so that

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

and our plane equation becomes :

$$g(\vec{X}) = \vec{W}^T\vec{X}$$

Least squares estimation of a hyperplane

A popular method for estimating a hyperplane is to compute a least-squares estimate using matrix algebra. This method provides a direct, closed form solution.

Assume a training set of M training samples $\{\vec{X}_m\}$ with indicator variables $\{y_m\}$ such that $y_m=+1$ for class 1 and $y_m = -1$ for class 2.

Our goal is to determine a discriminant function $g(\vec{X}) = \vec{W}^T\vec{X} + b$

For convenience we will add a "0th" term to X and W, so that

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

and our hyperplane is expressed as $g(\vec{X}) = \vec{W}^T\vec{X}$

We seek the "best" $\vec{W}$. This can be determined by minimizing a "Loss" function that can be defined as the Square of the error.

$$L(\hat{W}) = \sum_{m=1}^{M} (y_m - \vec{X}_m^T\hat{W})^2$$

To build or function, we will use the M training samples to compose a matrix **X** and a vector **Y**.

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

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix} \text{ (M coefficients).}$$

We can factor the loss function to obtain: $L(W) = (\mathbf{Y} - \mathbf{X}^T W)^T (\mathbf{Y} - \mathbf{X}^T W)$

To minimize the loss function, we calculate the derivative and solve for W when the derivative is 0.

$$\frac{\partial L(W)}{\partial W} = -2\,\mathbf{XY} + \mathbf{2\,X\,X}^{\mathrm{T}}\mathrm{W} = 0$$

which gives $\mathbf{XY} = \mathbf{X\,X}^{\mathrm{T}}\,\mathrm{W}$

and thus $\mathrm{W} = (\,\mathbf{XX}^{\mathrm{T}}\,)^{-1}\ \mathbf{X\,Y}$

An unknown event $\vec{X}$ can then be classified as

if $\vec{W}^T\vec{X} > 0$ then $\hat{\omega}_1$ else $\hat{\omega}_2$

A Committee of Boosted Classifiers

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.

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

With a committee of linear classifiers, each classifier "votes" on a target detection.

We can sum the votes for n classifiers with with $\sum_{n=1}^{N} \mathrm{sgn}(\vec{W}_n^T\vec{X})$

For a committee of N classifiers, the decision rule is:

if $\sum_{n=1}^{N} \mathrm{sgn}(\vec{W}_n^T\vec{X}) > 0$ Class 1 (True) else class 2 (False).

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

The sum of the biased votes would be $\sum_{n=1}^{N} \mathrm{sgn}(\vec{W}_n^T\vec{X} + B)$

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

if $\sum_{n=1}^{N} \mathrm{sgn}(\vec{W}_n^T\vec{X} + B) > 0$ Class 1 (True) else class 2 (False).

Learning a Committee of Classifiers with Boosting

We can iteratively learn a committee of classifiers using boosting. For this we will create a diagonal matrix A_i of "weights" a^i_m for each training sample. Initially, at i=0, all the weights are 1.

For each cycle, i, the classifier $\vec{W}_i$ and be learned from

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

where A_i = diagonal(a^i_m) is a diagonal matrix whose diagonal elements are a^i_m

Alternatively, we can select the "best" feature from among the single feature classifiers

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

given the boosted training data. The best classifier, W_d is the classifier that gives the most correct classifications for the boosted data.

$$\vec{W}_i = \arg-\max_d\left\{\sum_{m=1}^{M}(a^i_m y_m \cdot \vec{W}_d^T X_m)\right\}$$

After each new classifier is added, we recalculate the weights to give more weight to improperly classified training samples.

As we add classifiers, whenever a sample is miss-classified by the committee we will increase its weight so that it carries more weight in the next classifier added.

For m = 1 to M: if ($y_m \cdot \sum_{i=1}^{I} \mathrm{sgn}(\vec{W}_i^T \vec{X}_m)) < 0$ then $a^{i+1}_m \leftarrow a^i_m + 1$

The result is the (i+1)th weight vector
Set $i \leftarrow i+1$, and A^i_m = Diagonal(a^i_m)

We then learn the next classifier using the re-weighted data.

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

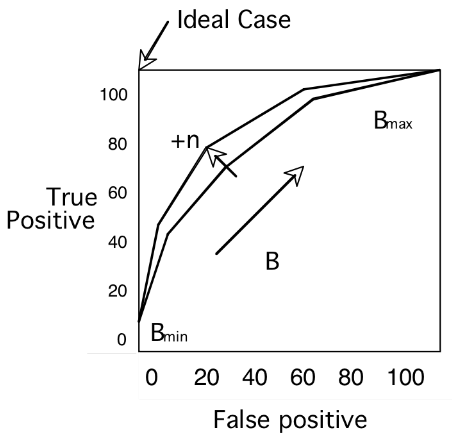
or

$$\vec{W}_i = \arg-\max_{d}\left\{\sum_{m=1}^{M}(a_m^i y_m \cdot \vec{W}_d^T X_m)\right\}$$

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 Boosting theorem states that adding a new boosted classifier 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.



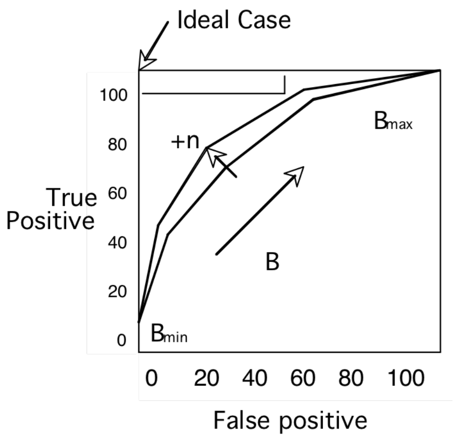
However, in general, the improvement provided for each new classifier becomes progressively smaller. We can end with a very very large number of classifiers.

Note that the probability of error can be computed from the FPR and FNR

p(Error) = #FP + #FN.

Learning a Multi-Stage Cascade of Classifiers

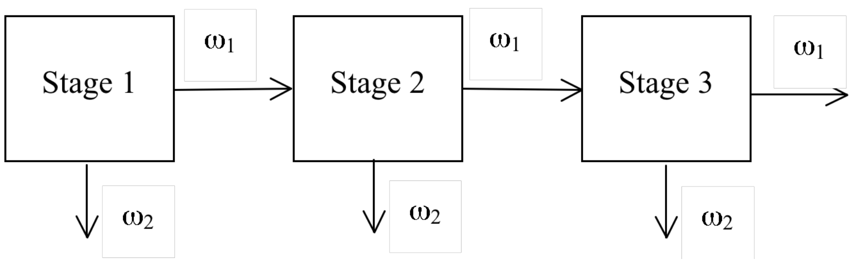
We can optimize the computation time by using a multi-stage cascade.
We can use a bias B to construct a committee that is biased to have avoid missing true positives (high True Positive Rate) at the cost of accepting many False Positives (high False Positive Rate).



We can the construct a second stage that is designed to eliminate the false positives that pass the first stage. We can the construct a second stage that is designed to eliminate the false positives that pass the first stage.

We can repeat this idea to construct multiple stages where each stage eliminates the easy True Negatives, while accepting all True Positives and many false positives.

False positives are then eliminated by later stages. Later stages are more expensive but they are called used less often.



Stages are organized so that each committee is successively more costly and more discriminant.

Assume a set of M training samples $\{X_m\}$ with labels $\{y_m\}$.
Set a desired error rate for each stage n : (FPR_n, FNR_n).

For each stage, S, train the S+1 stage with all positive samples from the previous stage.

Each stage acts as a filter, rejecting a grand number of easy 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.

Because we know the error rates for each committee we can estimate the probability of a detection based on the number of stages that an observation passes.