CISC 889 Bioinformatics (Spring 2004)

Support Vector Machines I

The metholodogy

Terminologies

- An object **x** is represented by a set of m attributes x^i, $1 \leq i \leq m$.
- A set of n training examples $S = \{ (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where y_i is the classification (or label) of instance $\mathbf{x}_i$.
  - For binary classification, $y_i = \{-1, +1\}$, and for k-class classification, $y_i = \{1, 2, \ldots, k\}$.
  - Without loss of generality, we focus on binary classification.
- The task is to learn the mapping: $\mathbf{x}_i \rightarrow y_i$
- A machine is a learned function/mapping/hypothesis h:

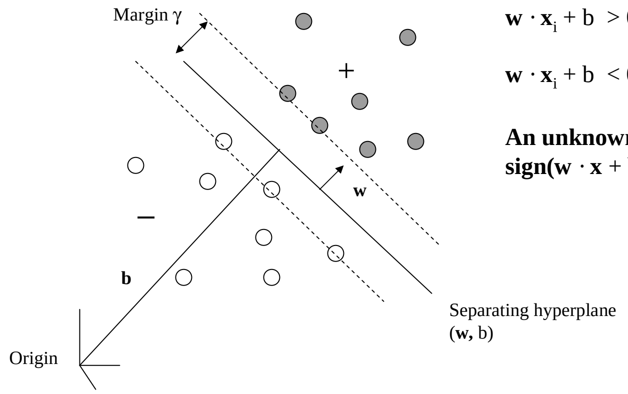
$$\mathbf{x}_i \rightarrow h(\mathbf{x}_i, \alpha)$$

where α stands for parameters to be fixed during training.

- Performance is measured as

$$E = (1/2n)\sum_{i=1 \text{ to } n} |y_i - h(\mathbf{x}_i, \alpha)|$$

Linear SVMs: find a hyperplane (specified by normal vector **w** and perpendicular distance **b** to the origin) that separates the positive and negative examples with the largest margin.



$\mathbf{w} \cdot \mathbf{x}_i + b > 0$ if $y_i = +1$

$\mathbf{w} \cdot \mathbf{x}_i + b < 0$ if $y_i = -1$

An unknown x is classified as sign(w · x + b)

Rosenblatt's Algorithm (1956)

η; // is the learning rate
$w_0 = \mathbf{0}$; $b_0 = 0$; $k = 0$
$R = \max_{1 \le i \le n} \| x_i \|$

error = 1; // flag for misclassification/mistake
while (error) { // as long as modification is made in the for-loop
    error = 0;

    for (i = 1 to n) {
        if (y_i (<$\mathbf{w}_k \cdot \mathbf{x}_i$> + b_k) ≤ 0){ // misclassification
            $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta\, y_i\, \mathbf{x}_i$ // update the weight
            $b_{k+1} = b_k + \eta\, y_i\, R^2$ // update the bias
            k = k +1
            error = 1;
        }
    }
}
return ($\mathbf{w}_k$, b_k) // hyperplane that separates the data, where k is the number of // modifications.

Questions w.r.t. Rosenblatt's algorithm

- Is the algorithm guaranteed to converge?
- How quickly does it converge?

Novikoff Theorem:

Let S be a training set of size n and $R = \max_{1 \le i \le n} \| x_i \|$. If there exists a vector w* such that $\|w^*\| = 1$ and

$$y_i\, (\mathbf{w^*} \cdot \mathbf{x}_i) \ge \gamma,$$

for $1 \le i \le n$, then the number of modifications before convergence is at most

$$(R/\gamma)^2.$$

Proof:

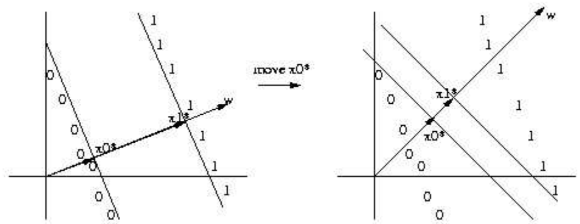
1. $\mathbf{w}_t \cdot \mathbf{w}^* = \mathbf{w}_{t-1} \cdot \mathbf{w}^* + \eta\, y_i\, \mathbf{x}_i \cdot \mathbf{w}^* \ge \mathbf{w}_{t-1} \cdot \mathbf{w}^* + \eta\, \gamma$
   $\mathbf{w}_t \cdot \mathbf{w}^* \ge t\, \eta\, \gamma$
2. $\| \mathbf{w}_t \|^2 = \| \mathbf{w}_{t-1} \|^2 + 2\, \eta\, y_i\, \mathbf{x}_i \cdot \mathbf{w}_{t-1} + \eta^2 \| \mathbf{x}_i \|^2$
   $\le \| \mathbf{w}_{t-1} \|^2 + \eta^2 \| \mathbf{x}_i \|^2$
   $\le \| \mathbf{w}_{t-1} \|^2 + \eta^2 R^2$
   $\| \mathbf{w}_t \|^2 \le t\, \eta\, R^2$
3. $\sqrt{t}\, \eta\, R\, \|\mathbf{w}^*\| \ge \mathbf{w}_t \cdot \mathbf{w}^* \ge t\, \eta\, \gamma$
   $t \le (R/\gamma)^2.$

Note:

- Without loss of generality, the separating plane is assumed to pass the origin, i.e., no bias b is necessary.
- The learning rate η seems to have no bearing on this upper bound. (why?)
- What if the training data is not linearly separable, i.e., w* does not exist?

Larger margin is preferred:

- converge more quickly
- generalize better



Dual form

- The final hypothesis w is a linear combination of the training points:

$$\mathbf{w} = \sum_{i=1 \text{ to } n} \alpha_i \, y_i \mathbf{x}_i$$

  where α_i are positive values proportional to the number of times misclassification of $\mathbf{x}_i$ has caused the weight to be updated.
- Vector $\boldsymbol{\alpha}$ can be considered as alternative representation of the hypothesis; α_i can be regarded as an indication of the information content of the example $\mathbf{x}_i$.
- The decision function can be rewritten as

  $h(x) = \text{sign}\,(\mathbf{w} \cdot \mathbf{x} + b)$

  $= \text{sign}((\sum_{j=1 \text{ to } n} \alpha_j \, y_j \mathbf{x}_j) \cdot \mathbf{x} + b)$

  $= \text{sign}(\sum_{j=1 \text{ to } n} \alpha_j \, y_j \, (\mathbf{x}_j \cdot \mathbf{x}) + b)$

Rosenblatt's Algorithm in dual form

$\boldsymbol{\alpha} = \mathbf{0}$; b = 0

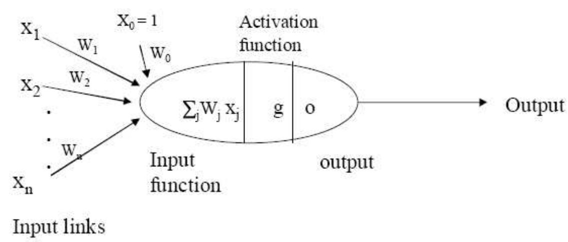
R = max $_{1 \le i \le n}$ $\| x_i \|$

```
error = 1; // flag for misclassification
while (error) {  // as long as modification is made in the for-loop
    error = 0;
    for (i = 1 to n) {
        if (y_i (Σ_{j=1 to n} α_j y_j (x_j · x_i) + b) ≤ 0 ){     // misclassification
                α_i = α_i +  1      // update the weight
                b = b +  y_i R^2   // update the bias
                error = 1;
        }
    }
}
return (α, b)     // hyperplane that separates the data, where k is the number of
                  // modifications.
```

Notes:

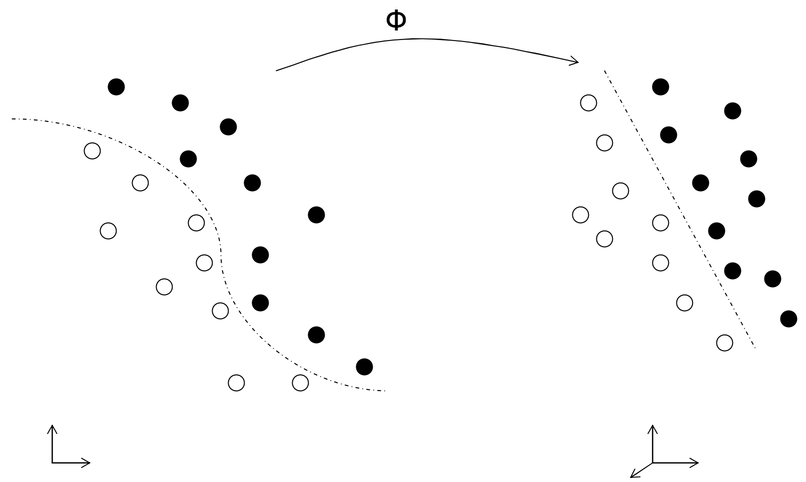
- The training examples enter the algorithm as dot products $(\mathbf{x}_j \cdot \mathbf{x}_i)$.
- α_i is a measure of information content; $\mathbf{x}_i$ with non-zero information content ($\alpha_i > 0$) are called support vectors, as they are located on the boundaries.

Relationship to linear perceptrons



- Linear SVMs are almost identical to linear perceptrons
- They differ from each other when are generalize to handle non linear cases.

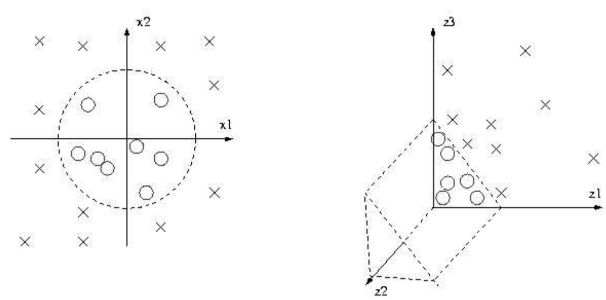
Non-linear mapping to a feature space



Nonlinear SVMs

Input Space Feature Space

$$\mathbf{x} \longrightarrow \Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$



Kernel function for mapping

- For input $\mathbf{X}= (x_1, x_2)$, Define map $\Phi(\mathbf{X}) = (x_1x_1, \sqrt{2}x_1x_2, x_2x_2)$.
- Define Kernel function as $K(\mathbf{X},\mathbf{Y}) = (\mathbf{X}\cdot\mathbf{Y})^2$.
- It has $K(\mathbf{X},\mathbf{Y}) = \Phi(\mathbf{X}) \cdot \Phi(\mathbf{Y})$
- **We can compute the scalar product in feature space without computing Φ.**

$$K(\mathbf{X},\mathbf{Y}) = \Phi(\mathbf{X}) \cdot \Phi(\mathbf{Y})$$
$$= (x_1 x_1, \sqrt{2}\, x_1x_2, x_2 x_2) \cdot (y_1 y_1, \sqrt{2}\, y_1y_2, y_2 y_2)$$
$$= (x_1x_1y_1y_1 + 2x_1x_2y_1y_2 + x_2x_2y_2y_2)$$
$$= (x_1y_1 + x_2y_2)(x_1y_1 + x_2y_2)$$
$$= ((x_1, x_2) \cdot (y_1, y_2))^2$$
$$= (\mathbf{X}\cdot\mathbf{Y})^2$$

Mercer's condition

Since kernel functions play an important role, it is important to know if a kernel gives dot products (in some higher dimension space).

For a kernel K(x,y), if for any g(x) such that $\int g(x)^2\, dx$ is finite, we have

$$\int K(x,y)g(x)g(y)\, dx\, dy \geq 0,$$

then there exist a mapping Φ such that

$$K(x,y) = \Phi(x) \cdot \Phi(y)$$

Notes:

1. Mercer's condition does not tell how to actually find Φ.
2. Mercer's condition may be hard to check since it must hold for every g(x).

More kernel functions

some commonly used generic kernel functions

- Polynomial kernel: $K(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}\cdot\mathbf{y})^p$
- Radial (or Gaussian) kernel: $K(\mathbf{x},\mathbf{y}) = \exp(-\|x-y\|^2/2\sigma^2)$

Questions: By introducing extra dimensions (sometimes infinite), we can find a linearly separating hyperplane. But how can we be sure such a mapping to a higher dimension space will generalize well to unseen data? Because the mapping introduces flexibility for fitting the training examples, how to avoid overfitting?

Answer: Use the maximum margin hyperplane. (Vapnik theory)

$\mathbf{w} \cdot \mathbf{x}_+ + b = +1$

$\mathbf{w} \cdot \mathbf{x}_- + b = -1$

$\gamma = ½ [(\mathbf{x}_+ \cdot \mathbf{w}/\|\mathbf{w}\|_2) - (\mathbf{x}_- \cdot \mathbf{w}/\|\mathbf{w}\|_2)]$

$= 1/\|\mathbf{w}\|_2$

Therefore, maximizing the geometric margin γ is equivalent to minimizing $\|w\|_2$, under linear contraints.

$\text{Min}_{\mathbf{w},b} < \mathbf{w} \cdot \mathbf{w} >$

subject to $y_i <\mathbf{w} \cdot \mathbf{x}_i> +b \geq 1$ for $i = 1, \ldots, n$

Lagrangian Theory

Quadratic programming optimization problem

… guaranteed to converge to the global minimum because of its being a convex

Note: advantages over the artificial neural nets

Advanced Issues

- Soft margin
  - Allow misclassification, but with penalties
- Multiclass classification
  - Indirect: combine multiple binary classifiers into a single multiclass classifier
  - Direct: generalize binary classification methods
- SVM Regression
- Support vector clustering by Ben-Hur et al (2001)

References and resources

- Cristianini & Shawe-Tayor, "*An introduction to Support Vector Machines*", Cambridge University Press, 2000.
- Chris Burges, A tutorial
- www.kernel-machines.org
- SVMLight