5. Linear Discriminant Functions

By

Ishwarryah S Ramanathan

Nicolette Nicolosi

Agenda

- 5.5 Minimizing Perceptron Criterion Function
  - The Perceptron Criterion Function
  - Convergence Proof for Single Sample Correction
  - Direct Generalizations
- 5.6 Relaxation Procedures
  - The Descent Algorithm
  - Convergence proof
- 5.7 Non-Separable Behavior

5.5 Minimizing the Perceptron Criterion Function

5.5.1 The perceptron criterion function

- Construct a criterion function for solving linear equalities $a^t y_i > 0$.
- The naive approach:
  - Let $J(a; y_1, \ldots, y_n)$ = number of samples misclassified by a.
  - This is a poor approach because the function is piecewise constant.

The perceptron criterion function contd.

- A better choice is the **Perceptron criterion function**:
  - $J_p(a) = \Sigma_{y \in \gamma} -a^t y$
  - $\gamma(a)$ are the samples misclassified by a
  - J_p is 0 if no samples are misclassified

Comparing Criterion Functions

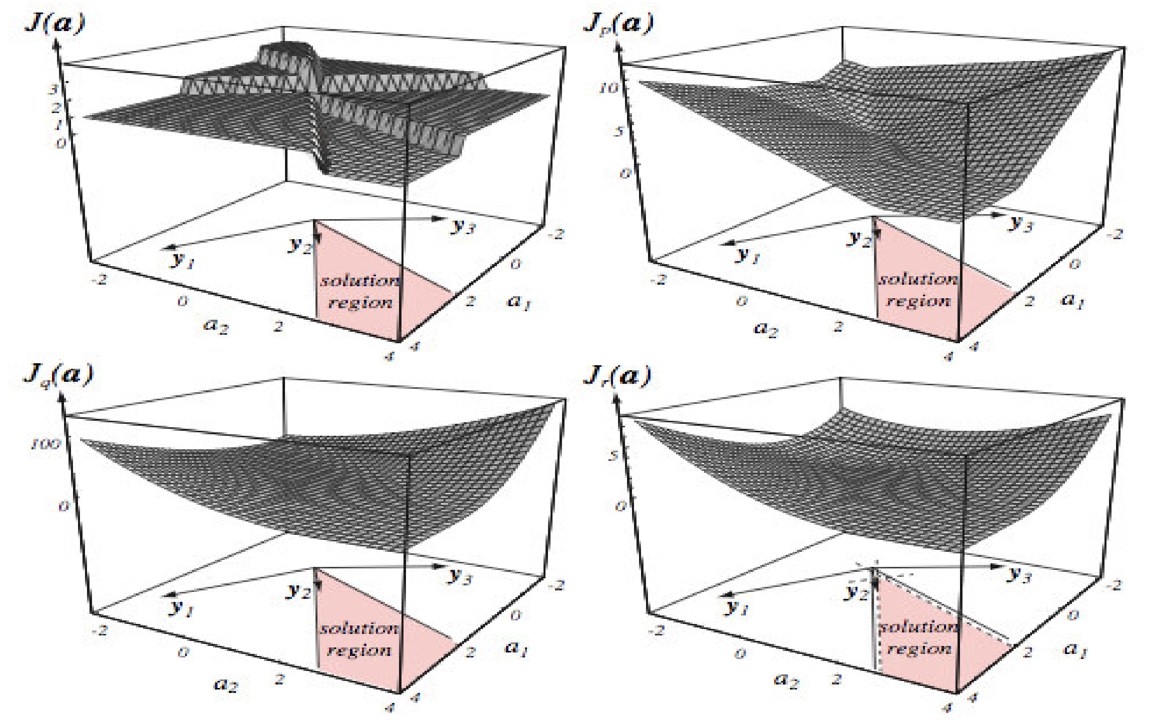

FIGURE 5.11. Four learning criteria as a function of weights in a linear classifier. At the upper left is the total number of patterns misclassified, which is piecewise constant and hence unacceptable for gradient descent procedures. At the upper right is the Perceptron criterion (Eq. 16), which is piecewise linear and acceptable for gradient descent. The lower left is squared error (Eq. 32), which has nice analytic properties and is useful even when the patterns are not linearly separable. The lower right is the square error with margin (Eq. 33). A designer may adjust the margin b in order to force the solution vector to lie toward the middle of the $b = 0$ solution region in hopes of improving generalization of the resulting classifier. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Constructing the Perceptron Algorithm

- The j-th component of the gradient of J_p is $\delta J_p / \delta a_j$. So,
  - $\Delta J_p = \Sigma_{y \epsilon \gamma} (-y)$
- This makes the update rule:
  - $a(k+1) = a(k) + \eta(k) \Sigma_{y \epsilon \gamma k} (y)$
  - γ_k is the set of samples misclassified by $a(k)$

The Batch Perceptron Algorithm

```
begin
init a, $\eta(*)$, criterion $\theta$, $k = 0$
do
    $k = k + 1$
    $a = a + \eta(k)\Sigma_{y \in \gamma_k}(y)$
until $|\eta(k)\Sigma_{y \in \gamma_k}(y)| < \theta$
return a
end
```

The Batch Perceptron Algorithm contd.

- Basically, the next weight vector is determined by adding the current weight vector to a multiple of the number of misclassified samples.
- The term **batch** is used because a large number of samples are involved in computing each update.
- Next slide: two-dimensional example with $a(1) = 0$ and $\eta(k) = 1$.

Perceptron Algorithm

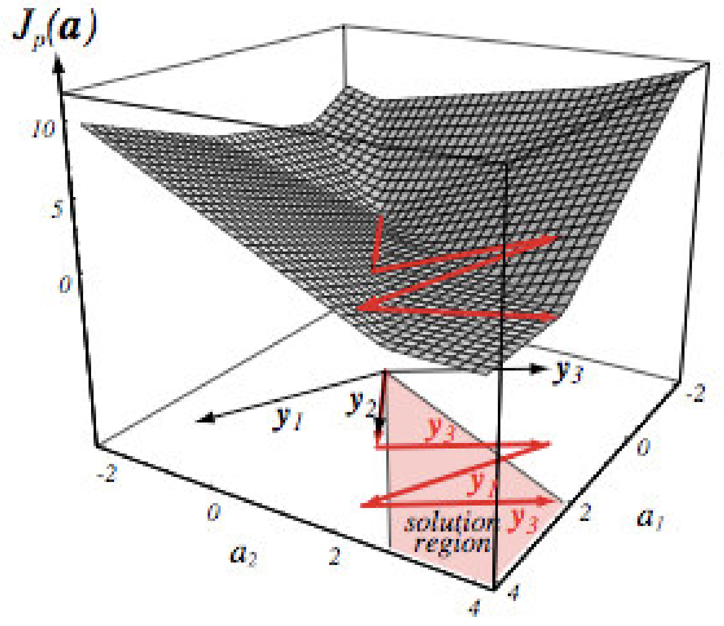

FIGURE 5.12. The Perceptron criterion, $J_p(\mathbf{a})$, is plotted as a function of the weights a_1 and a_2 for a three-pattern problem. The weight vector begins at $\mathbf{0}$, and the algorithm sequentially adds to it vectors equal to the "normalized" misclassified patterns themselves. In the example shown, this sequence is $\mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_1, \mathbf{y}_3$, at which time the vector lies in the solution region and iteration terminates. Note that the second update (by $\mathbf{y}_3$) takes the candidate vector *farther* from the solution region than after the first update (cf. Theorem 5.1). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

5.5.2 Convergence Proof of Single-Sample Correction

Single Sample Perceptron

- Some simplifying assumptions:
  - Modify the weight vector whenever a single sample in the sequence is misclassified
  - $\eta(k)$ is constant - there is a fixed increment
  - y^k is a sample; y_k is a misclassified sample
  - This results in the **fixed-increment rule** for generating a sequence of weight vectors:

▪ $a(1)$	arbitrary
▪ $a(k+1) = a(k) + y^k$,	where $k >= 1$

Fixed-Increment Single Sample Perceptron Algorithm

```
begin
init a, $k$ = 0
do
   $k$ = ($k$ + 1) mod $n$
     if $y^k$ is misclassified by a
     then a = a + $y^k$
until all patterns classified
return a
end
```

Fixed-Increment Rule

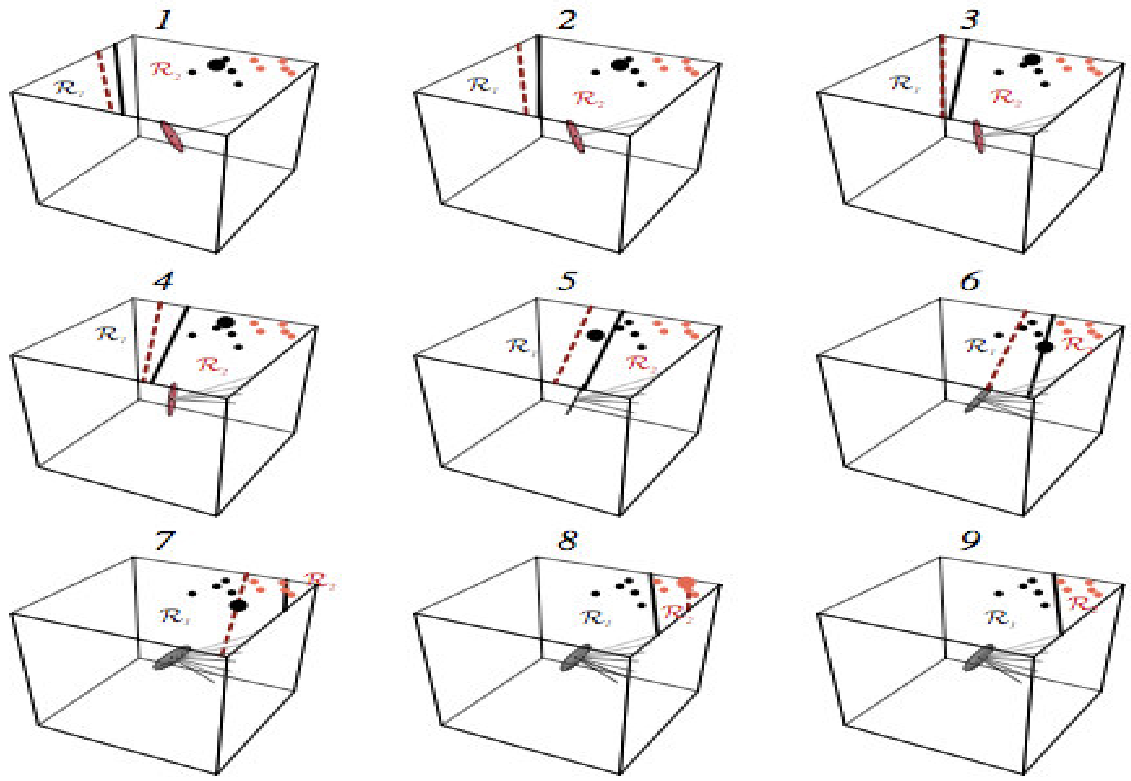

FIGURE 5.13. Samples from two categories, ω_1 (black) and ω_2 (red) are shown in augmented feature space, along with an augmented weight vector **a**. At each step in a fixed-increment rule, one of the misclassified patterns, $\mathbf{y}^k$, is shown by the large dot. A correction $\Delta\mathbf{a}$ (proportional to the pattern vector $\mathbf{y}^k$) is added to the weight vector—toward an ω_1 point or away from an ω_2 point. This changes the decision boundary from the dashed position (from the previous update) to the solid position. The sequence of resulting **a** vectors is shown, where later values are shown darker. In this example, by step 9 a solution vector has been found and the categories are successfully separated by the decision boundary shown. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Perceptron Convergence

- If samples are linearly separable, then the previous algorithm will succeed and return a solution vector.
- Idea:
  - If a_2 is any solution vector, then $||a_2(k+1) - a_2||$ is smaller than $||a(k) - a_2(k)||$
  - True for many solution vectors, but not in general (see steps 6 and 7 in previous slide).

Perceptron Convergence contd.

- We've already seen $a(k+1) = a(k) + y^k$. a_2 is any solution vector such that $a_2^t y_i$ is positive for all i. We can add a scale factor α and rewrite this equation:

$$a(k+1) - \alpha a_2 = (a(k) - \alpha a_2) + y^k$$

$$||a(k+1) - \alpha a_2||^2 = ||a(k) - \alpha a_2||^2 + 2(a(k) - \alpha a_2)^t y^k + ||y^k||^2$$

- Because y^k was misclassified, $a^t(k)yk <= 0$, giving:

$$||a(k+1) - \alpha a_2||^2 <= ||a(k) - \alpha a_2||^2 + 2\alpha a_2^t y^k + ||y^k||^2$$

Perceptron Convergence contd.

- Because $a_2^t y^k$ is strictly positive, the second term will overpower the third term if α is large enough.
- For example, let β be the maximum pattern vector length ($\beta = \max_i \|\, y_i \,\|^2$), and γ be the smallest inner product of the solution vector with any pattern vector ($\gamma = \min_i [a_2^t y_i] > 0$).

Perceptron Convergence contd.

- We now have the inequality:
  $||a(k+1) - \alpha a_2||^2 <= ||a(k) - \alpha a_2||^2 - 2\alpha\gamma + \beta^2$
- Choosing $\alpha = \beta^2/\gamma$ gives:
  $||a(k+1) - \alpha a_2||^2 <= ||a(k) - \alpha a_2||^2 - \beta^2$
- So, the squared distance between a(k) and αa_2 is reduced by at least β^2 after each correction. After k corrections, we have:
  $||a(k+1) - \alpha a_2||^2 <= ||a(1) - \alpha a_2||^2 - k\beta^2$

Perceptron Convergence contd.

- The squared distance can not be negative, so there must be at most k_0 corrections, where $k_0 = ||a(1) - \alpha a_2||^2 / \beta^2$.
- The weight vector that results after the corrections are done must classify all samples correctly, because there are a finite number of corrections that only occur at each misclassification.
- In summary, k_0 is a bound on the number of corrections, and there will always be a finite number of corrections using the fixed-increment rule if the samples are linearly separable.

5.5.3 Direct Generalizations

Variable Increment Perceptron

- A variant that uses a variable increment $\eta(k)$ and a margin b, and corrects when $a^t(k)y^k$ does not exceed the margin.
- The update function is now:

$$a(1) \qquad \text{arbitrary}$$
$$a(k+1) = a(k) + \eta(k)y^k \qquad k >= 1$$

where $a^t(k)y^k <= b$ for all k

Variable-Increment Perceptron Convergence

- It can be shown that a(k) converges to a solution vector a satisfying $a^t y_i > b$ for all i if the samples are linearly separable and if:

$$\eta(k) >= 0$$

$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta(k) = \infty \qquad \lim_{m\to\infty} \frac{\sum_{k=1}^{m} \eta^2(k)}{\left(\sum_{k=1}^{m} \eta(k)\right)^2} = 0$$

- These conditions on η(k) are satisfied if η(k) is a positive constant or decreases as 1/k.

Variable-Increment Perceptron with Margin

```
begin
init a, threshold $\Theta$, margin $b$, $\eta(*)$, $k$ = 0
do
   $k = (k + 1) \bmod n$
     if $a^t y^k <= b$
     then $a = a + \eta(k) y^k$
until $a^t y^k > b$ for all $k$
return a
end
```

Batch Variable Increment Perceptron

- Another variant is the original gradient descent algorithm for J_p. The update function for this is:
  $a(1)$ arbitrary
  $a(k+1) = a(k) + \eta(k) + \Sigma_{y \in \gamma k}\, y \quad , \quad k >= 1$
- It can be easily shown to return a solution. If a_2 is a solution vector, it correctly classifies the correction vector:
  $$y^k = \Sigma_{y \in \gamma k}\, y$$

Batch variable Increment Perceptron contd.

```
begin
init a, $\eta(*)$, $k = 0$
do
    $k = (k + 1) \bmod n$
    $\gamma_k = \{\}$
    $j = 0$
      do $j = j + 1$
        if $y_j$ is misclassified
        then append $y_j$ to $\gamma_k$
      until $j = n$
      $a = a + \eta(k)\Sigma_{y \cdot \gamma_k}(y)$
until $\gamma_k = \{\}$
return a
end
```

Balanced Winnow Algorithm

- Related to the Perceptron algorithm - the main difference is that the Perceptron algorithm returns a weight vector with components a_i ($i = 0, \ldots, d$), and the Winnow algorithm scales these by $2\sinh[a_i]$.
- The balanced Winnow algorithm has separate positive (a^+) and negative (a^-) weight vectors that are each associated with one of two categories to be learned. Corrections to either only occur if a training pattern in its own set is misclassified.

Balanced Winnow Algorithm contd.

```
begin
init $a^+$, $a^-$, $\eta(*)$, $k = 0$, $\alpha > 1$
if Sgn[$a^{+t}y_k - a^{-t}y_k$] != $z_k$ //misclassified
then
    if $z_k = +1$
   then $a_i^+ = \alpha^{+y_i}a_i^+$; $a_i^- = \alpha^{-y_i}a_i^-$ for all i
    if $z_k = -1$
    then $a_i^+ = \alpha^{-y_i}a_i^+$; $a_i^- = \alpha^{+y_i}a_i^-$ for all i
return $a^+$, $a^-$
end
```

Balanced Winnow Algorithm contd.

- The main benefits of this version over the Perceptron algorithm are:
  - The two weight vectors move in a uniform direction and the "gap" between them never increases. This leads to a more general convergence proof than that of the Perceptron.
  - Convergence is generally faster. This is because each weight does not go past the final value if the learning rate is correctly set.

5.6 Relaxation Procedures

5.6.1 The Descent Algorithm

- We have the perceptron criterion function
  - $Jp(a) = \Sigma\ y \in \breve{Y}\ (-a^t y)$ . . . Eq 1
  - Ў is the set of training samples misclassified by ‘a’, the solution vector
- Relaxation Procedure
  - ‘Relaxation’ is a generalized approach to minimize the perceptron criterion function (eq 1) by linear classification.
  - Jp(.) is a function with ‘a’ as a solution vector. Similar function Jq(.). Both denotes the misclassified sample.

The Descent Algorithm contd.

- $J_q(a) = \Sigma_{y \in \breve{y}} (a^t y)^2$ . . . Eq 2
- Adv: The curve is smoother (search is easier) and continuous
- There are two problems with this criterion
  - The function is too smooth and slowly converge to a=0
  - J_q is dominated by training samples with large magnitude

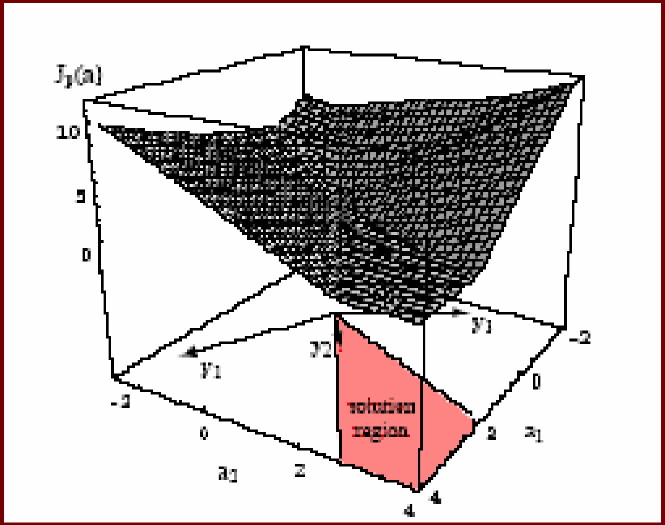

Fig.1 $J_p(a)$

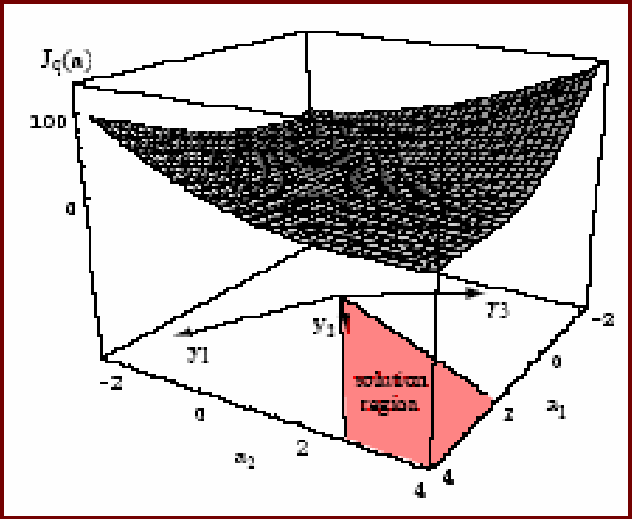

Fig.2 $J_q(a)$

[Taken from 'Pattern Classification' from Duda and Hart]

The Descent Algorithm contd.

- To avoid the problems of J_q, we choose other perceptron function J_r.
  - $J_r = ½ \Sigma_{y є ў} ((a^t y-b)^2 / |y|^2)$ . . . Eq 3

    where $Ў(a)$ are samples of $a^t y <= b$
  - $J_r(a)$ is never negative
  - $J_r(a) = 0$, $a^t y >= b$ for all training samples or when $Ў$ is empty
  - Gradient $= \delta J_r(a) / \delta a = \Delta J_r$
    $= ½ . 2 \Sigma_{y є ў} ((a^t y-b).y)/ |y|^2$
    $= \Sigma_{y є ў} ((a^t y-b).y)/ |y|^2$
  - Therefore,
    $a(k+1) = a(k) + \eta(k) \Sigma_{y є ў} ((a^t y-b).y)/ |y|^2$
    where $\eta(k)$ is learning rate that sets the step size

Batch Relaxation with Margin

```
begin
init a, $\eta(*)$, b, k = 0
do k=(k+1) mod n
     $Y_k$ = {}
     j = 0
     do j = j+1
          if $a^t y^j <= b$ then append $y^j$ to $Y_k$
     until j = n
     a = a+ $\eta(k) \Sigma_{y \in Y} ((b-a^t y).y) / |y|^2$
until $Y_k$ = {}
return a
end
```

Relaxation Algorithm contd.

- By Single-Sample Correction Rule analogous to Eq 3,
  - a(1), arbitrary
  - **$a(k+1) = a(k) + \eta\, ((b-a^t(k).y^k).y^k) \,/\, |y^k|^2$** . . . Eq 4
    where $a^t(k).y^k <= b$ for all k

Single-Sample Relaxation with Margin

```
begin
init a, $\eta(*)$, b, k = 0
do k=(k+1) mod n
    if $a^t y^k$ <= b
    then a=a+ $\eta(k).((b-a^t(k).y^k).y^k)/|y^k|^2$
until $a^t y^k$ > b for all $y^k$
return a
end
```

Single Sample Relaxation Rule

- Geometric Interpretation of Single-Sample Relaxation Rule
  - $r(k) = (b - a^t(k).y^k) / |y^k|$
  - $r(k)$ is the distance from the hyperplane $a^t y^k = b$ to $a(k)$.
  - $y^k / |y^k|$ is the unit normal vector of hyperplane

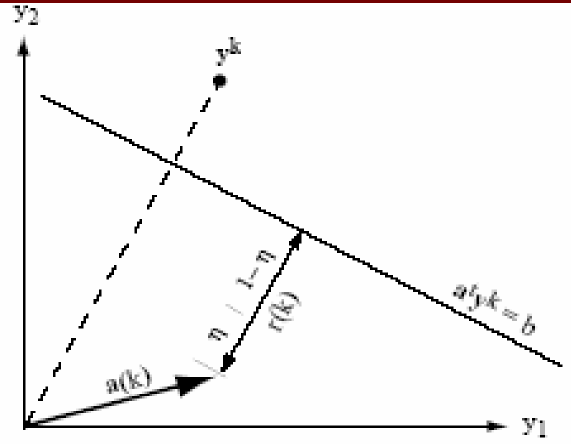

Figure 5.14: In each step of a basic relaxation algorithm, the weight vector is moved a proportion η of the way towards the hyperplane defined by $\mathbf{a}^t\mathbf{y}^k = b$.

Geometric Interpretation of Single-Sample Relaxation Rule

- If $\eta=1$, $a(k)$ is moved to the hyperplane
- Eq 4 becomes,

$$a^t(k+1).y^k-b = (1-\eta)\,(a^t(k).y^k - b)$$

- Under-Relaxation
  - If $\eta<1$ then $a^t(k+1).y^k$ is less than b
  - Slow descent or failure to converge
- Over-Relaxation
  - If $\eta>1$ then $a^t(k+1).y^k$ is greater than b
  - Overshoots, but convergence will finally be achieved
- Note: We restrict $0 < \eta < 2$

5.6.2 Convergence Proof

Convergence Proof

- Relaxation rule is on a set of Linearly Separable Sample. The corrections may not be finite
- As the number of corrections goes to infinity we could see a(k) converges to a limit vector on the boundary of the solution region.
- We can say region of $a^t y >= b$ is in the region of $a^t y>0$ if $b>0$
- Which implies that a(k) will enter this region

Convergence Proof contd.

- Proof
  - Say ậ is any vector in solution region
  - ie., $ậ^t y^i > b$ for all i then, a(k) gets closer to ậ at each step
  - From Eq 4,
    $||a(k+1) - ậ||^2 = ||a(k) - ậ||^2 - 2\eta ((ậ-a(k))^t y^k) (b-a^t(k).y^k)/||y^k||^2$
    $+ \eta^2 (b-a^t(k).y^k)2/||y^k||^2$
  - And $(ậ-a(k))^t y^k > b-a^t(k)y^k >= 0$
  - So,
    $||a(k+1) - ậ||^2 <= ||a(k) - ậ||^2 – \eta(2 - \eta) (b-a^t(k)y^k)^2 / ||y^k||^2$
  - Because $0<\eta<2$,
    $$||a(k+1) - ậ|| <= ||a(k) - ậ||$$
  - Thus a(1),a(2),… gets closer to ậ and as limit k goes to infinity, ||a(k) - ậ|| approaches a limiting distance r(ậ)

Convergence Proof contd.

  - That is, $a(k)$ confines to a hyper-sphere of radius $r(\hat{a})$ with $\hat{a}$ as centre, as k goes to ∞.
  - Therefore, Limiting $a(k)$ is confined to intersect hyper-spheres centered about all of the possible solution vectors.
- The intersection of the hyper-spheres is a single point on the boundary of the solution region.
  - Say a' and a'' are on the common intersection, then $\|a' - \hat{a}\|=\|a'' - \hat{a}\|$ for every $\hat{a}$ in the solution region.
  - But $\|a' - \hat{a}\|=\|a'' - \hat{a}\|$ implies that the solution region is contained in $(d-1)$ dimensional hyper-plane of points equidistant from a' and a''.
  - But the solution region is d dimensional.
  - Which implies $a(k)$ converges to point 'a'.

Convergence Proof contd.

- The point ‘a’ is not inside the solution region because then the sequence would be finite.
- It is not outside the region, because the correction will make ‘a’ move η times its distance from boundary plane.
- Therefore, *the limit point must be on the boundary*.

5.7 Non-separable behavior

- The Perceptron and Relaxation procedures are methods for finding a separating vector when the samples are linearly separable. They are error correcting procedures
- Separating Vector in training does not mean the resulting classifier will perform well on independent test data
- For performance, many training samples should be used
- Sufficiently large training samples are almost certainly not linearly separable
- No weight vector can correctly classify every sample in a non-separable set

5.7 Non-separable Behavior contd.

- Number of corrections in the Perceptron and Relaxation procedures goes to infinity if set is non-separable
- Performance improves if $\eta(k) \to 0$ as $k \to \infty$
- The rate at which $\eta(k)$ approaches zero is important:
  - Too slow: Results will be sensitive to those training samples that render the set non-separable.
  - Too fast: Weight vector may converge prematurely with less than optimal results.

Research Ideas

- Paper by Khardon and Wachman discusses performance of variants of the Perceptron algorithm.
  - They discovered that the variant that uses a margin has the unintended side effect of tolerating noise and stabilizing the algorithm.
  - The authors implemented several variants of the algorithm and performed experiments using different sets of data to compare performance.
  - Good overview of a practical implementation and its results.

Research Ideas

- Paper by S. J. Wan on Cone Algorithm – Extension to Perceptron Algorithm
  - Perceptron convergence is important in machine learning algorithms
  - Find a covering cone for a finite set of linearly contained vectors and the convergence
  - Theorems, definitions and equations are well defined.

References

- Pattern Classification, 2nd ed by Duda R, Hart P, Stork D, 2001. John Wiley & Sons.
- Noise Tolerant Variants of the Perceptron Algorithm by R. Khardon and G. Wachman - Journal. Mach. Learn. Res. Pg 227-248 May2007
- Cone Algorithm: An Extension of the Perceptron Algorithm by S. J. Wan - IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 10, October 1994

THANKS!