Morso

5.2 Linear Discriminant Functions and Decision Surfaces

A discriminant function that is a linear combination of the components of x can be written as

$$g(x) = w^T x + \omega_0$$

w: the weight vector; ω_0: bias or threshold weight

5.2.1 The two category case.

(a) The underlying decision rule is

decide ω_1 if $g(x) > 0$ and ω_2 if $g(x) \leq 0$ (* arbitrary)

$$w^T x + \omega_0 > 0$$

$$w^T x > -\omega_0 \Rightarrow \omega_1$$

$$\text{or} \quad \omega_2$$

(b)

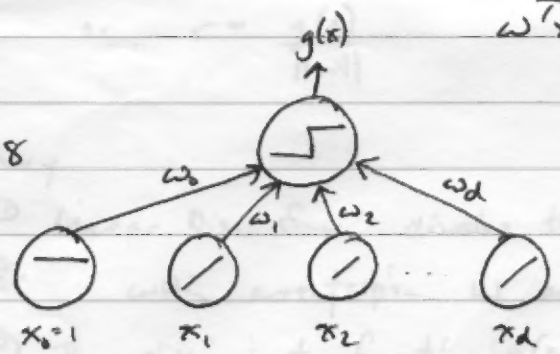


Fig 5.18

(c) $g(x) = 0$ defines the decision surface that separates the classes

→ Hyperplane for the linear case

For two points x_1 and x_2 on the decision surface,

$$w^T x_1 + \omega_0 = w^T x_2 + \omega_0 \Rightarrow w^T (x_1 - x_2) = 0$$

w is orthogonal to any vector in the hyperplane.

$g(x) > 0 \Rightarrow \omega_1$ or R_1 → "positive" side 5.2 #2

ω_2 or R_2 → "negative" side

Horse

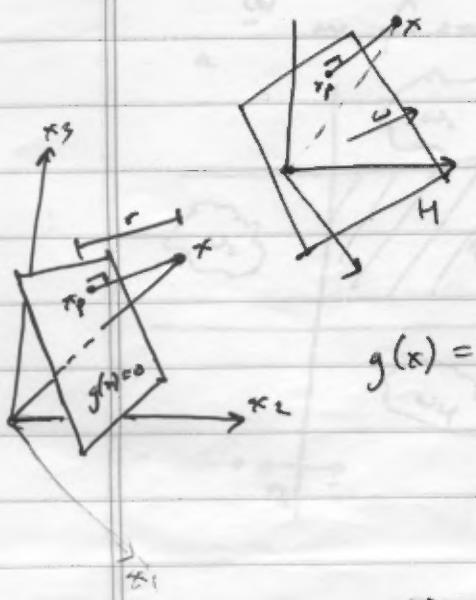
(d) d-func. $g(x)$ computes and algebraic distance from x to the hyperplane

H_w is the hyperplane (given by w and ω_0)

$$x = x_p + r\frac{w}{\|w\|}$$

x_p: normal projection of x onto H_w

r: the algebraic distance. r is positive on the pos. side and negative on the neg.



(*) $g(x_p) = w^T x_p + \omega_0 = 0$

$w^T x_p = -\omega_0$

$$g(x) = w^T x + \omega_0$$

$$w^T\left(x_p + r\frac{w}{\|w\|}\right) + \omega_0$$

$$w^T x_p + r\frac{w^T w}{\|w\|} + \omega_0$$

$$-\omega_0 + r\|w\| + \omega_0$$

$$g(x) = r\|w\|$$

or $$r = \frac{g(x)}{\|w\|}$$

Note the distance of the plane to the origin is $\frac{\omega_0}{\|w\|}$

(e) Summary

① Linear Disc. func. divides the feature space by a hyperplane decision surface with orientation w and location by ω_0

② The discriminant function $g(x)$ is proportional to the signed distance from x to the hyperplane.

$g(x) > 0 \longrightarrow$ positive side

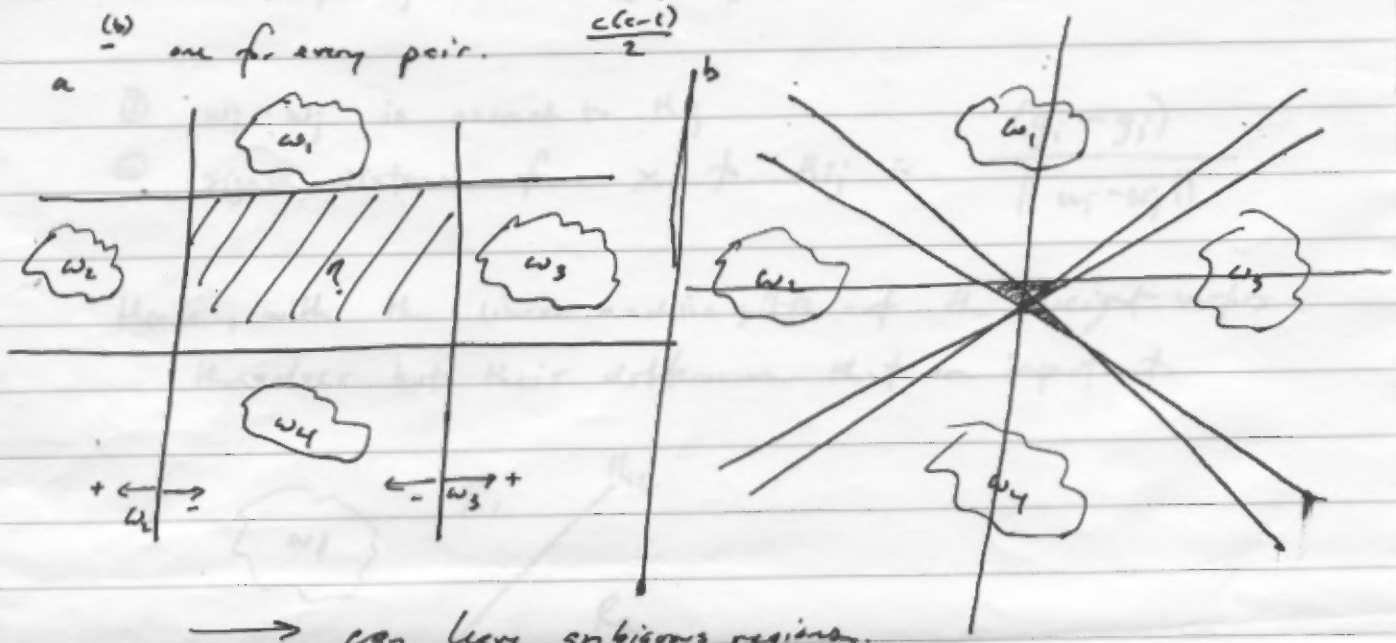
$g(x) < 0 \longrightarrow$ negative side.

5.2.2. Multicategory Case

For c-classes discriminants, we can devise multiple ways to use linear discriminants.

(a) c leave-one-out classifiers.

(b) one for every pair. $\frac{c(c-1)}{2}$



→ can leave ambiguous regions.

- maybe this problem at ↑ can be avoided —
- Def. Linear Machine.

  define c linear discriminant functions

$$g_i(x) = w_i^T x + w_{i0} \quad , \quad i = 1, \ldots, c$$

assign x to ω_i if $g_i(x) > g_j(x)$ for all $j \neq i$.

for ties, classification is undefined

- Result: divides feature space into c decision regions

Recall $g_i(x) = w_i^T x + w_{i0}$

- For contiguous R_i & R_j, boundary is a portion of a hyperplane H_{ij}, defined by $g_i(x) = g_j(x)$

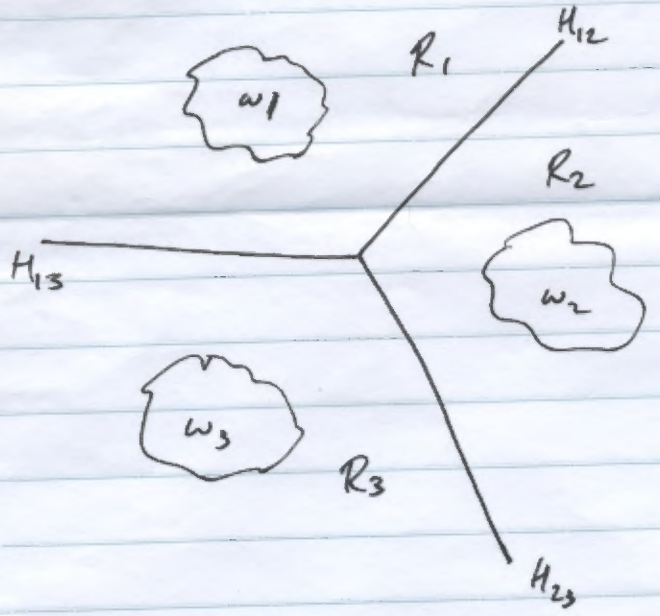
or

$$(w_i - w_j)^T x + (w_{i0} - w_{j0}) = 0$$

① $w_i - w_j$ is normal to H_{ij}

② signed distance from x to H_{ij} is $\dfrac{(g_i - g_j)}{\| w_i - w_j \|}$

Hence, with the linear machine, it is not the weight vectors themselves but their differences that are important.



③ Decision Regions for a linear machine are convex.

— limiting the flexibility & accuracy of the classifier

Plots

- Linear Disc. Function $g(x)$ can also be written as

$$g(x) = w_0 + \sum_{i=1}^{d} w_i x_i$$

- Quadratic Discriminant Function

$$g_2(x) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j$$

Additional $\frac{d(d+1)}{2}$ coefficients $(w_{ij} = w_{ji})$

$g(x)=0$ defines hyperquadric decision surface.
recall the general multivariate Gaussian.

- Polynomial Discriminant Functions

$$g_3(x) = g_2(x) + \sum_{i=1}^{d}\sum_{j}\sum_{k} w_{ijk} x_i x_j x_k$$

and so on...

- More generally, these can be thought of as truncated series expansions of some $g(x)$

Def. Generalized Linear Discriminant

$$g(x) = \sum_{i=1}^{\hat{d}} a_i y_i(x) \quad \text{or} \quad g(x) = a^T y$$

($y_i(x) = \phi_i(x)$; $y = \phi(x)$)

$\hat{d}$ $\phi_i(x)$ are arbitrary functions of x. (e.g. feature detection) $\Big\}$ $\hat{d}$-dim weight vector

Flamo

5.8 #2

- smart selection of $\phi(x)$ we can approximate any discriminant function by such an expansion
- Resulting discriminant is not linear in x by it is linear in $\phi(x)$

Simple Example : Homogeneous Coordinates

For linear $g(x) = w_0 + \sum_{i=1}^{d} w_i x_i$

$$= \sum_{i=0}^{d} w_i x_i \quad \text{with } x_0 = 1$$

$$\phi \; y = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} 1 \\ x \end{bmatrix} \Bigg\} \text{ augmented feature vector}$$

$$a = \begin{bmatrix} w_0 \\ w_i \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} w_0 \\ w \end{bmatrix} \Bigg\} \text{ augmented weight vector.}$$

Although trivial, it is quite convenient

- reduces solution to only weight vector a
- preserves all distances among samples
- the ϕ vector lie in a d-dim subspace (the x-space) and the H passes through origin of ϕ-space $a^T \phi(x) = 0$ but lie anywhere in x-space (translation).

Morse

5.3 #3

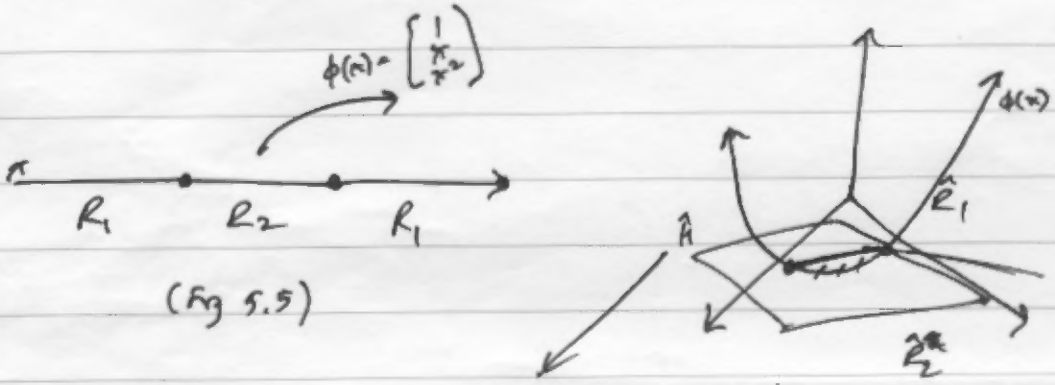
Quadratic Example

$d=1$

$$g(x) = a_1 + a_2 x + a_3 x^2$$

$$\hookrightarrow \quad \phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$$

The data remain inherently one-dimensional → varying x causes $\phi(x)$ to trace out a curve in 3 dimensions.



(Fig 5.5)

- $\hat{H}$ divides the ϕ-space into two decision regions $\hat{R}_1, \hat{R}_2$
- Decision Regions in ϕ-space are convex, but not necessarily in x-space.

  Generally, the decision surfaces in x-space can be complex.

- Curse of Dim often makes it hard to capitalize on this flexibility in practice. e.g. even for modest d, the $\hat{d}$ grows too big. $\frac{(d+1)(d+2)}{2}$ grows
- Clearly, a general series expansion of $g(x)$ can easily lead to unrealistic requirements on computation & data

↳ Later in the week, we will see how to use similar ideas but not fall into this problem

CSE 672 435312

1,2 are by & intro.

Morse

5.4 #1
5.5

5.4 Two Category Linearly Separable Case

5.5 Minimizing the Perceptron Criterion Function

Intro: We learned about discriminants; their representation, limitations and potentials. Now we want to discuss how to train them.

Recall: Our generalized linear discriminant setting: we have some ~~features~~ samples $y_1, \dots y_n$ (these are computed from our input data $x_1, \dots, x_n$ with the kernel function $\phi(\cdot)$).

Each sample has a label ω_1 or ω_2.

Need to use these samples to learn a of $g(x) = a^T\phi(x) = a^T y$

↳ y_i is classified correctly if $a^T y_i > 0$ for ω_1 & $a^T y_i < 0$ for ω_2

Def: Linearly Separable

Assume: (for the moment) That the data is linearly separable — that there exists a weight vector that classifies all of the samples correctly.

Normalize Data to simplify two-category case.

$$y_i = \begin{cases} +\phi(x_i) & \text{if } x_i \text{ label is } \omega_1 \\ -\phi(x_i) & \text{if } x_i \text{ label is } \omega_2 \end{cases}$$

Allows use to forget labels and look for a weight vector a such that

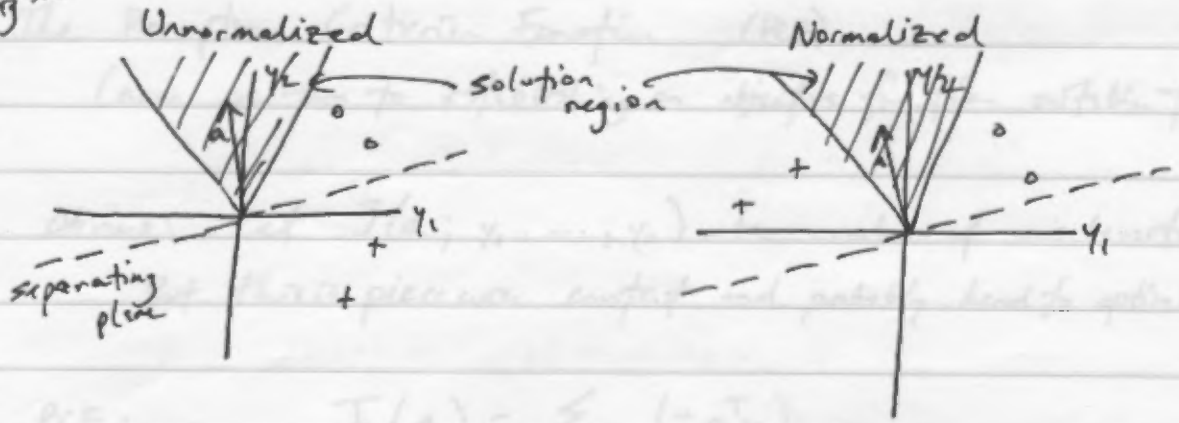
$$a^T y_i > 0 \text{ for all of the samples.}$$

Def. Such a weight vector is called a solution or separating vector.

Hasso

5.4
5.5 #2

Fig 5.8



Solution region defines the set of weight vectors that all perfectly separate the data samples.

① Solution Vector is not unique

↳ Constrain?

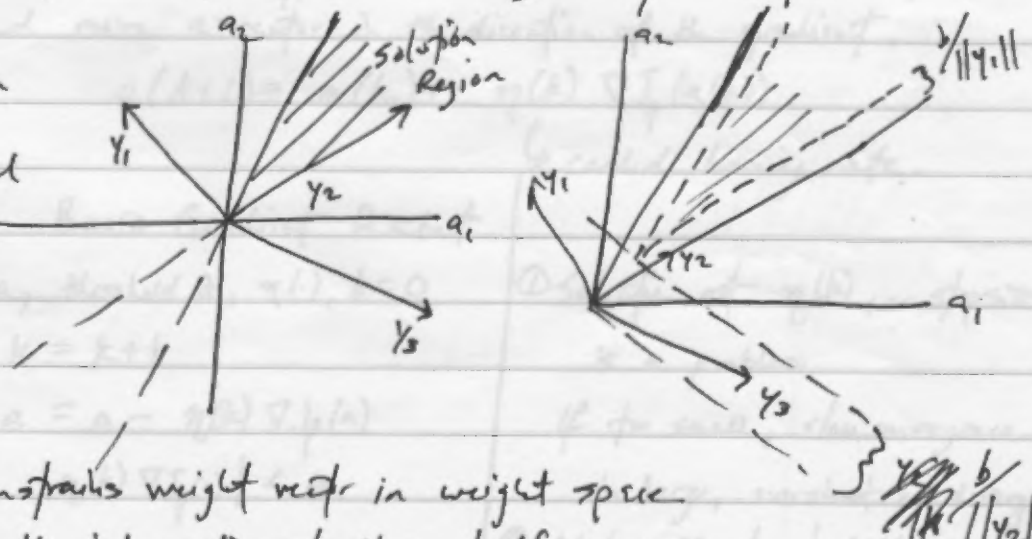
(i) seek a weight vector minimizing distance from samples to the separating plane

(ii) seek a minimum length weight vector satisfying

$$a^T y_i \geq b \quad \forall i \text{ given a } b > 0$$

↑ margin

The margin constrains solution region by $\frac{b}{\|y_i\|}$



Motivation: margin pushes solution toward the middle of the solution region ⓑ it is natural to assume this will generalize to new test samples better. than one on the boundary.

Each sample constrains weight vector in weight space
Solution lies in the intersection of these half-spaces.

Kesso

5.4
5.5 #3

The Perceptron Criterion Function (PCF)
(now we turn to establishing an objective function suitable for finding a)

Obvious: Let $J(a; y_1, \ldots, y_n)$ be number of misclassified samples
But this is piecewise constant and probably hard to optimize.

PCF: $$J_p(a) = \sum_{y \in Y} (-a^T y)$$

↑ set of incorrectly classified points.

① If all are classified correctly, $Y = \phi$ and $J_p(a)$ is 0.

② $J_p(a)$ is non-negative
(If $a^T y_i > 0$, then $y_i \notin Y$)

Ask. ③ Geometrically, $J_p(a)$ is proportional to the sum of the distances from the misclassified samples to the decision boundary.

Gradient Descent:

- One suitable way of optimizing $J_p(a)$ is with Gradient Descent
- Idea: start with an arbitrary (why?) a-vector, compute gradient $\nabla J_p(a)$ and move a-vector in the direction of the gradient.

$$a(k+1) = a(k) + \eta(k) \nabla J_p(a(k))$$

↳ called learning rate

Algorithm Basic Gradient Descent

1 Input a, threshold θ, $\eta(\cdot)$, $k=0$
2 do $k = k+1$
3 $a = a - \eta(k) \nabla J_p(a)$
4 until $|\eta(k) \nabla J_p(a)| < \theta$
5 Return a.

① Selection of $\eta(k)$, or stepsize, is a problem.
If too small, slow convergence.
Too large, overshoot or diverge
② Note Newton & L-M.

Florso

54
515 #4

Optimizing PLF with Gradient Descent. (Batch Perceptron)

$$J_p(a) = \sum_{y \in Y} (-a^T y) \quad , \quad \nabla J_p = \sum_{y \in Y} (-y)$$

$$\Rightarrow \quad a(k+1) = a(k) + \eta(k) \sum_{y \in Y_k} y$$

$Y_k \rightarrow$ misclassified set. by $a(k)$

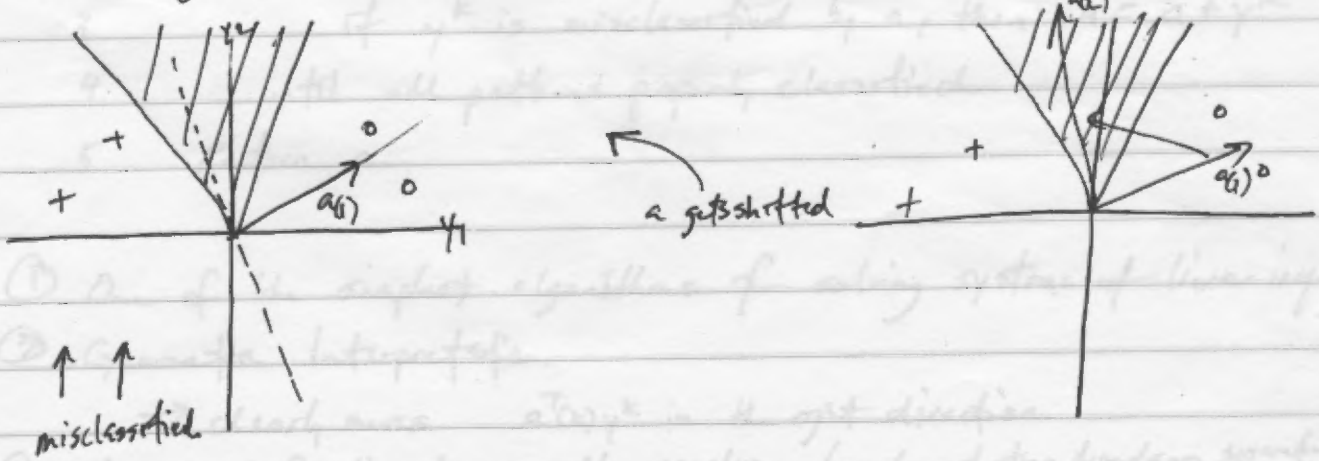
Algorithm Batch Perceptron

1 Input a, threshold θ, $\eta(\cdot)$, $k=0$
2 do $k = k+1$
3 $a = a + \eta(k) \sum_{y \in Y_k} y$
4 until $|\eta(k) \sum_{y \in Y_k} y| < \theta$
5 Return a

① Modify / Upate the weight vector by adding some multiple of the sum of the misclassified samples to the present weight vector

② Batch means that a large group of samples are used when updating

③ Convergent if the linearly separable problem (shown next in Single version)



Kerso

5.4
5.5 #5

- Fixed Increment Single-Sample Perceptron

$\eta(k)$ is constant. — $\eta(k)$ just scales samples.

Notation Change: in perceptron only the misclassified samples count.
denote the sequence of them $y^1, y^2, \cdots$
each y^k is one of the n samples $y_1, \cdots, y_n$

E.g. we consider the samples cyclically

$$\underset{\downarrow}{y_1}, \underset{\downarrow}{y_2}, \underset{\downarrow}{y_3}, \underset{\downarrow}{y_1}, \underset{\downarrow}{y_2}, y_3, y_1, \underset{\downarrow}{y_2}, y_3, \cdots$$

$$y^1, y^2, y^3, y^4, \quad y^5, \cdots$$

Fixed-increment rule is $a(1)$ arbitrary

$$a(k+1) = a(k) + y^k \qquad k \geq 1$$

where $a^T(k)\, y^k \leq 0 \quad \forall\, k$

Algorithm Fixed-Increment Single Sample Perceptron

```
1   Input   a, k=0
2     do    k=(k+1) mod n
3           If y^k is misclassified by a, then a = a + y^k
4     until all patterns properly classified.
5   Return a.
```

① One of the simplest algorithms for solving systems of linear inequalities.
② Geometric Interpretation.
→ clearly moves $a^T(k) y^k$ in the right direction.
③ Convergent for linearly separable samples (number of steps depends on separation of classes)

Flores

5.4
5.5 #6

Variable Increment Perceptron With Margin

Introduce Margin (as before)

$a(1)$ arbitrary

$a(k+1) = a(k) + \eta(k) y^k \quad k \geq 1$

where $a^T(k) y^k \leq b \quad \forall k.$

Algorithm VIPwM

1 Input a, ~~[illegible]~~, margin b, $\eta(\cdot)$, $k = 0$

2 do $k = (k+1) \bmod n$

3 if $a^T y^k \leq b$ then $a = a + \eta(k) y^k$

4 until $a^T y^k > b \quad \forall k$

5 return a

① if $\eta(k) \geq 0$, $\lim_{n \to \infty} \sum_{k=1}^{n} \eta(k) = \infty$

, $\lim_{n \to \infty} \sum_{k=1}^{n} \eta^2(k) \Big/ \left(\sum_{k=1}^{n} \eta(k) \right)^2 = 0$

then $a(k)$ converges to the solution vector a satisfying $a^T y_i > b \quad \forall i$

→ these conditions are satisfied if $\eta(k)$ is a positive constant or decreases as $1/k$

5.6 Relaxation Procedures

Perceptron Criterion does not have a continuous gradient.
(Hence second-order optimization is not possible.)

Squared Error Criterion

$$J_q(a) = \sum_{y \in \mathcal{Y}} (a^T y)^2$$

① Gradient of J_q is smooth.
But ② It can converge to a point on the boundary, e.g. $a=0$
③ It's value can be dominated by long sample vectors.

$$J_r(a) = \frac{1}{2} \sum_{y \in \mathcal{Y}} \frac{(a^T y - b)^2}{\|y\|^2} \qquad \checkmark\ a^T y \geq b$$

① Again if $\mathcal{Y} = \phi$ $J_r(a) \doteq 0$
② $J_r(a)$ is never negative and 0 iff $a^T y \geq b\ \forall\ y$

$$\nabla_a J_r = \sum_{y \in \mathcal{Y}} \frac{a^T y - b}{\|y\|^2} y$$

Update rule $a(1)$ arbitrary

$$a(k+1) = a(k) + \eta(k) \sum_{y \in \mathcal{Y}} \frac{b - a^T y}{\|y\|^2} y$$

Morse

5.6 #2

Algorithm Single-Sample Relaxation with Margin

1 Input a, $\eta(\cdot)$, $k=0$, margin b

2 do $k = (k+1) \bmod n$

3 if $a^T y^k \leq b$ then $a = a + \eta(k) \dfrac{b - a^T y^k}{\|y^k\|^2} y^k$

4 Until $a^T y^k > b \ \forall \ y^k$

5 return a.

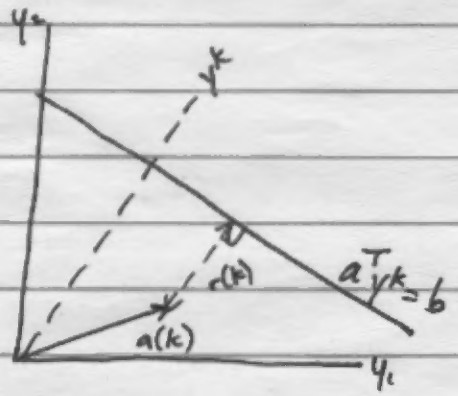
$r(k)$ some distance

Geometrical Interpretation.

$$r(k) = \frac{b - a^T(k) y^k}{\|y^k\|}$$

distance from $a(k)$ to the hyperplane $a^T y^k = b$

(b) $y^k/\|y^k\|$ is unit normal to the plane



Hence, the update moves $a(k)$ a ~~certain~~ fraction of the distance from the hyperplane.

If $\eta = 1$, $a(k)$ is moved exactly to the hyperplane, thereby "relaxing" the "tension" from $a^T y^k > b$

SSR+M is guaranteed to converge to at least a point on the boundary of the solution region.

Note — separable / non-separable behavior: these "error correcting" procedures will take an infinite number of iterations in the non-separable case.

Hons

5.8 Minimum Squared-Error

- Dealing with linear inequalities much less well understood than dealing with linear equalities.

- Past → make all $a^T y_i > 0$

  Now → make all $a^T y_i = b_i \quad \forall y_i$ and $b_i > 0$

$$Y \quad a = b$$

$$\begin{bmatrix} y_{10} & y_{11} & \cdots & y_{1d} \\ y_{20} & \cdots & & \\ \vdots & & & \\ y_{n0} & y_{n1} & \cdots & y_{nd} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

No exact solution exists for typical situations.

$$e = Ya - b$$

$$J_s(a) = \|Ya - b\|^2 = \sum_{i=1}^{n} (a^T y_i - b_i)^2$$

Minimize the squared length of the error vector which is equivalent to minimizing the sum of squared error criterion.

Closed form solution via pseudo-inverse

$$\nabla J_s = \sum_{i=1}^{n} 2(a^T y_i - b_i) y_i = 2 Y^T (Ya - b)$$

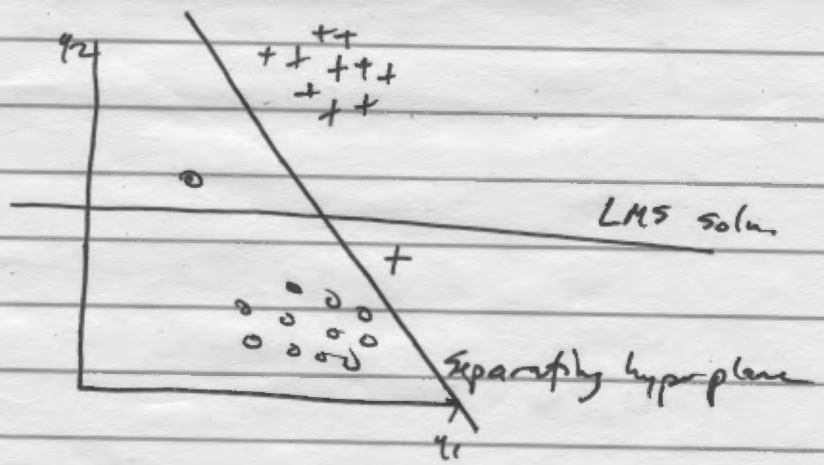
necessary condition $\underbrace{Y^T Y}_{\text{d×d and often non-singular}} a = Y^T b$

$$a = \underbrace{(Y^T Y)^{-1} Y^T}_{\text{pseudoinverse of } Y} b$$

closed form, unique MSE solution vector a.

MSE soln a depends on the margin vector, b.

* A reasonable b should give us a separating vector in the separable and non-separable case.



SVMs (Short Treatment)

Again, data is $x_1, \dots, x_n$.
Or with ϕ mapping, $y_1, \dots, y_n$.

Write $z_1, \dots, z_n$ to encode class of sample s.t.
$z_i = +1$ for class ω_1 and -1 for class ω_2.

GLD $g(y) = a^T y$

A separating hyperplane ensures $z_k g(y_k) \geq 1$ (*), $k = 1, \dots, n$
(for weight vector)

* Til now, optimization was based on a given and fixed b margin.
Here, in SVM, we want to determine the ~~margin~~ separating hyperplane that has the largest margin.

① Intuition: larger the margin, better the generalization.

(BG) → Recall distance from any hyperplane H_a is $\frac{|g(y)|}{\|a\|}$

$g(y) = a^T y$

$r \frac{a^T a}{\|a\|}$

$g(y) = r\|a\|$

$g(x_p) = w^T x_p + \omega_0 = 0$

$w^T x_p = -\omega_0$

$g(x) = w^T x + \omega_0$

$w^T\left(x_p + r\frac{w}{\|w\|}\right) + \omega_0$

$w^T x_p + r\frac{w^T w}{\|w\|} + \omega_0$

$g(x) = r\|w\|$

(*) implies

$$\frac{z_k g(y_k)}{\|a\|} \geq b, \quad k = 1, \dots, n$$

z_k: corrects sign; $\|a\|$: dist.; b: margin

① Solution vector can be scaled arbitrarily, so impose $b\|a\| = 1$, minimize $\|a\|^2$

$b = \frac{1}{\|a\|}$ (max, min)

Thm 3.2

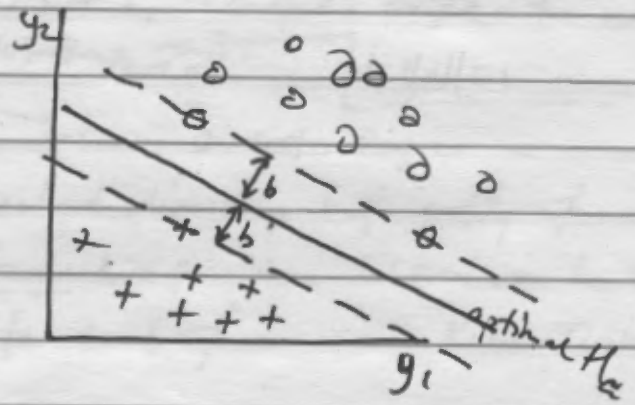
5.11 #2

Def. Support Vectors are the samples for which the eq. is an equality.

① Support vectors are all equally close to the hyperplane

② S.V. define the optimal separating hyperplane

③ S.V. are the most difficult to classify



Ask. Perceptron-like Training for SVM.

→ perceptron training updates the weight vector based on any misclassified pattern

→ for SVM, train by using the worst-classified pattern

① During training, such a pattern is one on the wrong side of the current decision boundary

② At the end of training, such a pattern will be one of the S.V.

Ask. However, Perceptron-SVM is impractical because finding the worst classified pattern can be computationally expensive.

Horse

5.11 #3

Standard SVM Training.
(Use Lagrange undetermined multipliers.)

$$L_p(a,\alpha) = \frac{1}{2}\|a\|^2 - \sum_{k=1}^{n} \alpha_k \left[z_k (a^T y_k + b) \right] + \sum_{k=1}^{n} \alpha_i$$

↑ constraints weighted with scale

$b\|a\| = 1$

C_1 ① Min $L(a,\alpha)$ wrt a, b
② Require all derivatives vanish wrt α_i
③ $\alpha_i \geq 0$ → Convex Quadratic Program.

Can be solved directly but costly if $\hat{d}$ is high. (C_2)

Convert to dual is possible. → maximize L subject to gradient of L wrt a vanishes and $\alpha_i \geq 0$

Convert to dual form via Kuhn-Tucker construction

$$L_D(\alpha) = \sum_{k=1}^{n} \alpha_i - \frac{1}{2} \sum_{k,j}^{n} \alpha_k \alpha_j z_k z_j y_j^T y_k$$

$$\text{s.t.} \quad \sum_{k=1} z_k \alpha_k = 0 \qquad \alpha_k \geq 0 \qquad k = 1, \ldots, n.$$

gives conditions

$\frac{\partial}{\partial a} a = \sum_{k=1}^{n} \alpha_k z_k y_k$ Solve with quadratic programming.

$\frac{\partial}{\partial b} \sum_{k=1}^{n} \alpha_k z_k = 0$

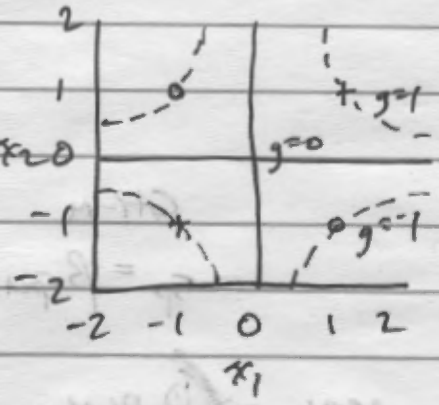
Notice here in dual form, the lifted data points themselves are not directly needed y_i... only the dot-products btw pairs of them. $\boxed{y_j^T y_k}$

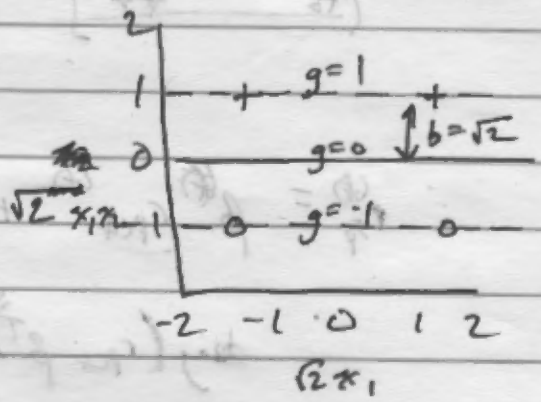
C_2 occurs at the same value as min. of L subject to C_1

↑ maximum of L subject to

Flores 5.11 #4

XOR Problem





XOR is simplest problem for which LD can't solve it.

Second order expansion, $1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2$

$\sqrt{2}$ gives convenient normalization

↓

Solution is $g(x) = x_1x_2$

Decision hyperplane is $g=0$.

Margin is $b = \frac{1}{\|a\|} = \sqrt{2}$.