Minimum Squared Error

C12

Today

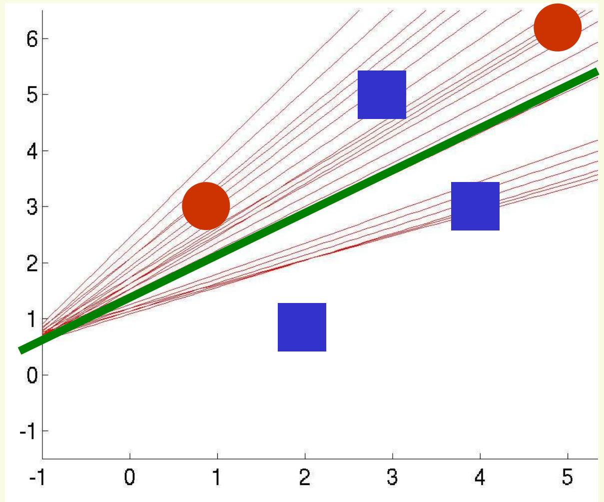
- Continue with Linear Discriminant Functions
  - Last lecture: Perceptron Rule for weight learning
  - This lecture: Minimum Squared Error (MSE) rule
    - Pseudoinverse
    - Gradient descent (Widrow-Hoff Procedure)
    - Ho-Kashyap Procedure

LDF: Perceptron Criterion Function

- The perceptron criterion function
  - try to find weight vector ***a*** s.t. $\boldsymbol{a}^t\boldsymbol{y}_i > 0$ for all samples $\boldsymbol{y}_i$
  - perceptron criterion function $J_p(\boldsymbol{a}) = \sum_{\boldsymbol{y} \in Y_M} \left(-\boldsymbol{a}^t\boldsymbol{y}\right)$
  - only look at the misclassified samples
  - will converge in the linearly separable case
- Problem:
  - will not converge in the nonseparable case
  - to ensure convergence can set

$$\eta^{(k)} = \frac{\eta^{(1)}}{k}$$

  - However we are not guaranteed that we will stop at a good point



LDF: Minimum Squared-Error Procedures

- Idea: convert to easier and better understood problem

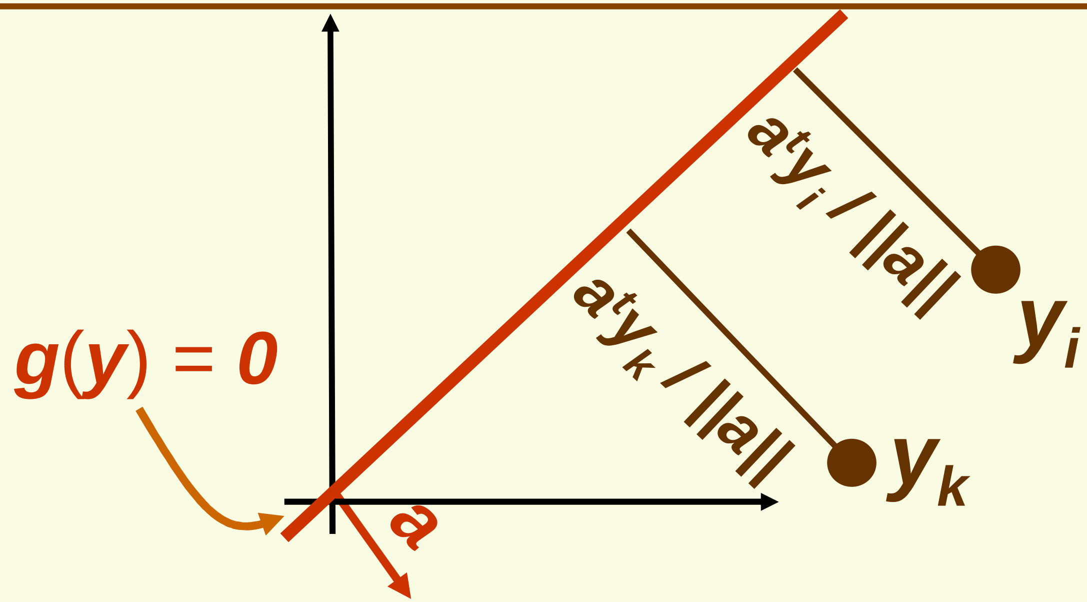
$\mathbf{a}^t\mathbf{y}_i > 0$ for all samples $\mathbf{y}_i$
solve system of linear inequalities

⇓

$\mathbf{a}^t\mathbf{y}_i = \mathbf{b}_i$ for all samples $\mathbf{y}_i$
solve system of linear equations

- MSE procedure
  - Choose **positive** constants $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$
  - try to find weight vector $\mathbf{a}$ s.t. $\mathbf{a}^t\mathbf{y}_i = \mathbf{b}_i$ for all samples $\mathbf{y}_i$
  - If we can find weight vector $\mathbf{a}$ such that $\mathbf{a}^t\mathbf{y}_i = \mathbf{b}_i$ for all samples $\mathbf{y}_i$, then $\mathbf{a}$ is a solution because $\mathbf{b}_i$'s are positive
  - consider all the samples (not just the misclassified ones)

LDF: MSE Margins



- Since we want $\boldsymbol{a}^t \boldsymbol{y}_i = \boldsymbol{b}_i$, we expect sample $\boldsymbol{y}_i$ to be at distance $\boldsymbol{b}_i$ from the separating hyperplane (normalized by ||***a***||)
- Thus $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n$ give relative expected distances or "margins" of samples from the hyperplane
- Should make $\boldsymbol{b}_i$ small if sample ***i*** is expected to be near separating hyperplane, and make $\boldsymbol{b}_i$ larger otherwise
- In the absence of any additional information, there are good reasons to set $\boldsymbol{b}_1 = \boldsymbol{b}_2 = \ldots = \boldsymbol{b}_n = \boldsymbol{1}$

LDF: MSE Matrix Notation

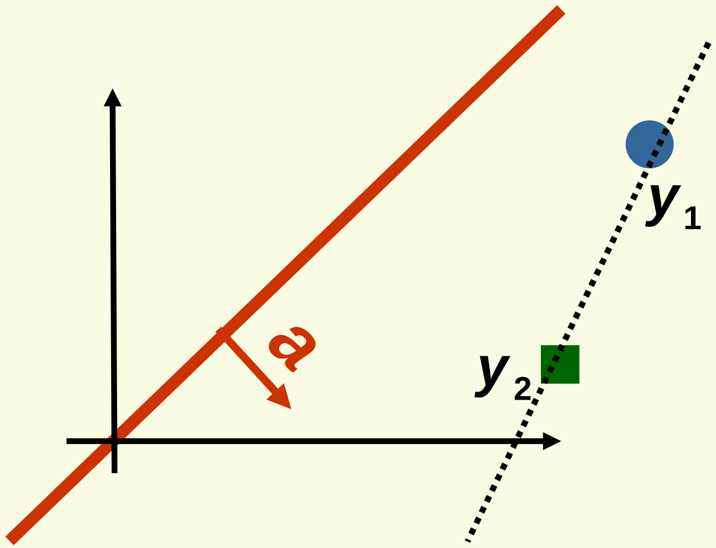
- Need to solve ***n*** equations $$\begin{cases} a^t y_1 = b_1 \\ \vdots \\ a^t y_n = b_n \end{cases}$$
- Introduce matrix notation:

$$\underbrace{\begin{bmatrix} y_1^{(0)} & y_1^{(1)} & \cdots & y_1^{(d)} \\ y_2^{(0)} & y_2^{(1)} & \cdots & y_2^{(d)} \\ \vdots & & & \vdots \\ y_n^{(0)} & y_n^{(1)} & \cdots & y_n^{(d)} \end{bmatrix}}_{Y} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix}}_{a} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{b}$$

- Thus need to solve a linear system ***Ya*** = ***b***

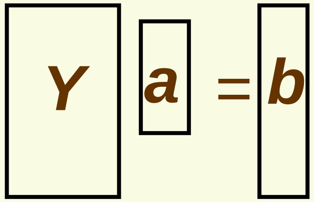
LDF: Exact Solution is Rare

- Thus need to solve a linear system ***Ya*** = ***b***
  - ***Y*** is an ***n*** by (***d +1***) matrix
- Exact solution can be found only if ***Y*** is nonsingular and square, in which case the inverse $\boldsymbol{Y}^{-1}$ exists
  - $\boldsymbol{a} = \boldsymbol{Y}^{-1}\boldsymbol{b}$
  - (number of samples) = (number of features + 1)
  - almost never happens in practice
  - in this case, guaranteed to find the separating hyperplane



LDF: Approximate Solution

- Typically ***Y*** is overdetermined, that is it has more rows (examples) than columns (features)
  - If it has more features than examples, should reduce dimensionality

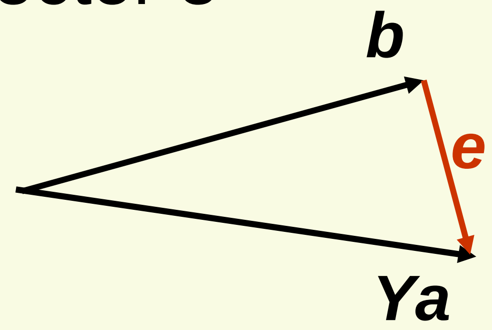


- Need ***Ya*** = ***b***, but no exact solution exists for an overdetermined system of equation
  - More equations than unknowns
- Find an approximate solution ***a***, that is ***Ya*** ≈ ***b***
  - Note that approximate solution ***a*** ***does not*** necessarily give the separating hyperplane in the separable case
  - But hyperplane corresponding to ***a*** may still be a good solution, especially if there is no separating hyperplane

LDF: MSE Criterion Function

- Minimum squared error approach: find ***a*** which minimizes the length of the error vector ***e***

$$\boldsymbol{e} = \boldsymbol{Ya} - \boldsymbol{b}$$



- Thus minimize the *minimum squared error* criterion function:

$$J_s(\boldsymbol{a}) = \|\boldsymbol{Ya} - \boldsymbol{b}\|^2 = \sum_{i=1}^{n} \left(\boldsymbol{a}^t \boldsymbol{y}_i - \boldsymbol{b}_i\right)^2$$

- Unlike the perceptron criterion function, we can optimize the minimum squared error criterion function analytically by setting the gradient to ***0***

LDF: Optimizing $J_s(a)$

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

- Let's compute the gradient:

$$\nabla J_s(a) = \begin{bmatrix} \dfrac{\partial J_s}{\partial a_0} \\ \vdots \\ \dfrac{\partial J_s}{\partial a_d} \end{bmatrix} = \frac{dJ_s}{da} = \sum_{i=1}^{n} \frac{d}{da}\left(a^t y_i - b_i\right)^2$$

$$= \sum_{i=1}^{n} 2\left(a^t y_i - b_i\right)\frac{d}{da}\left(a^t y_i - b_i\right)$$

$$= \sum_{i=1}^{n} 2\left(a^t y_i - b_i\right) y_i$$

$$= 2Y^t\left(Ya - b\right)$$

LDF: Pseudo Inverse Solution

$$\nabla J_s(\mathbf{a}) = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b})$$

- Setting the gradient to 0:

$$2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b}) = \mathbf{0} \;\Rightarrow\; \mathbf{Y}^t\mathbf{Ya} = \mathbf{Y}^t\mathbf{b}$$

- Matrix $\mathbf{Y}^t\mathbf{Y}$ is square (it has $d+1$ rows and columns) and it is often non-singular
- If $\mathbf{Y}^t\mathbf{Y}$ is non-singular, its inverse exists and we can solve for $\mathbf{a}$ uniquely:

$$\mathbf{a} = \boxed{(\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t}\,\mathbf{b}$$

pseudo inverse of $\mathbf{Y}$

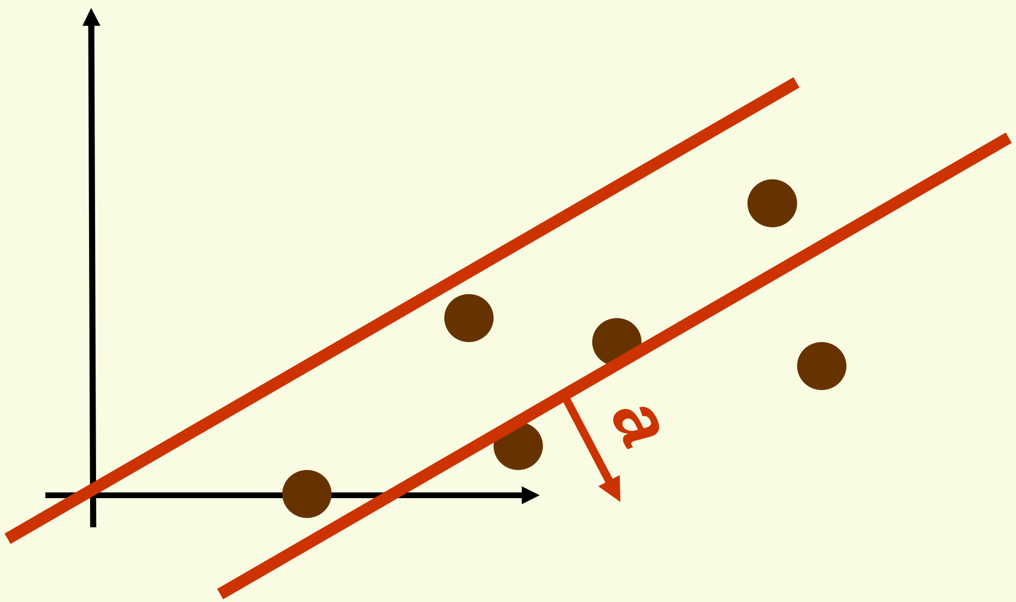
$$\left((\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t\right)\mathbf{Y} = (\mathbf{Y}^t\mathbf{Y})^{-1}(\mathbf{Y}^t\mathbf{Y}) = \mathbf{I}$$

LDF: Minimum Squared-Error Procedures

- If $\boldsymbol{b_1}=\ldots=\boldsymbol{b_n}=1$, MSE procedure is equivalent to finding a hyperplane of best fit through the samples $\boldsymbol{y_1},\ldots,\boldsymbol{y_n}$

$$J_s(\boldsymbol{a}) = \left\| \boldsymbol{Ya} - \boldsymbol{1_n} \right\|^2$$

$$\boldsymbol{1_n} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \Bigg\} n$$



- Then we shift this line to the origin, if this line was a good fit, all samples will be classified correctly

LDF: Minimum Squared-Error Procedures

- Only guaranteed the separating hyperplane if $\boldsymbol{Ya} > \boldsymbol{0}$
  - that is if all elements of vector $\boldsymbol{Ya} = \begin{bmatrix} \boldsymbol{a}^t\boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{a}^t\boldsymbol{y}_n \end{bmatrix}$ are positive
- We have $\boldsymbol{Ya} \approx \boldsymbol{b}$
- That is $\boldsymbol{Ya} = \begin{bmatrix} \boldsymbol{b}_1 + \varepsilon_1 \\ \vdots \\ \boldsymbol{b}_n + \varepsilon_n \end{bmatrix}$ where ε may be negative
  - If $\varepsilon_1,\dots,\varepsilon_n$ are small relative to $\boldsymbol{b}_1,\dots,\boldsymbol{b}_n$, then each element of $\boldsymbol{Ya}$ is positive, and $\boldsymbol{a}$ gives a separating hyperplane
  - If approximation is not good, ε_i may be large and negative, for some $\boldsymbol{i}$, thus $\boldsymbol{b}_i + \varepsilon_i$ will be negative and $\boldsymbol{a}$ is not a separating hyperplane
- Thus in linearly separable case, least squares solution ***a*** does *not necessarily* gives separating hyperplane
- But it will give a “reasonable” hyperplane

LDF: Minimum Squared-Error Procedures

- We are free to choose ***b***. May be tempted to make ***b*** large as a way to insure $\boldsymbol{Ya} \approx \boldsymbol{b} > 0$
- Does not work
  - Let β be a scalar, let's try $\beta\boldsymbol{b}$ instead of ***b***
  - if ***a**** is a least squares solution to ***Ya*** = ***b***, then for any scalar β, least squares solution to ***Ya*** = β***b*** is β***a****

$$\arg\min_{a}\|Ya - \beta b\|^2 = \arg\min_{a} \beta^2 \|Y(a/\beta) - b\|^2$$

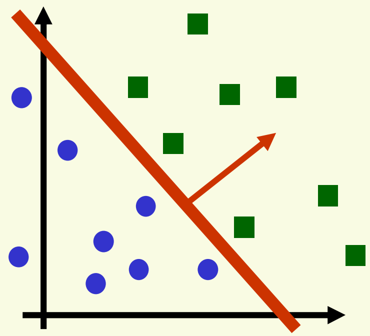
$$= \arg\min_{a}\|Y(a/\beta) - b\|^2 = \beta a^*$$

  - thus if for some ***i*** th element of ***Ya*** is less than 0, that is $y^t_i a < 0$, then $y^t_i(\beta a) < 0$,
- Relative difference between components of ***b*** matters, but not the size of each individual component

LDF: How to choose b in MSE Procedure?

- So far we assumed that constants $b_1, b_2, \ldots, b_n$ are positive but otherwise arbitrary
- Good choice is $b_1 = b_2 = \ldots = b_n = 1$. In this case,

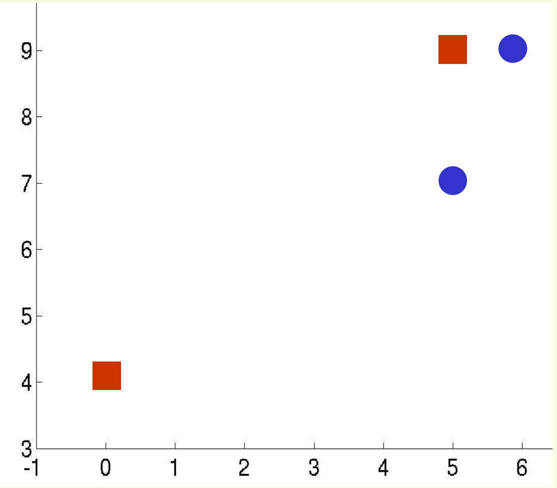
1. MSE solution is basically identical to Fischer's linear discriminant solution



2. MSE solution approaches the Bayes discriminant function as the number of samples goes to infinity

$$g_B(x) = P(c_1 \mid x) - P(c_2 \mid x)$$

LDF: Example



- Class 1: (6 9), (5 7)
- Class 2: (5 9), (0 4)
- Set vectors $\boldsymbol{y_1}$, $\boldsymbol{y_2}$, $\boldsymbol{y_3}$, $\boldsymbol{y_4}$ by adding extra feature and "normalizing"

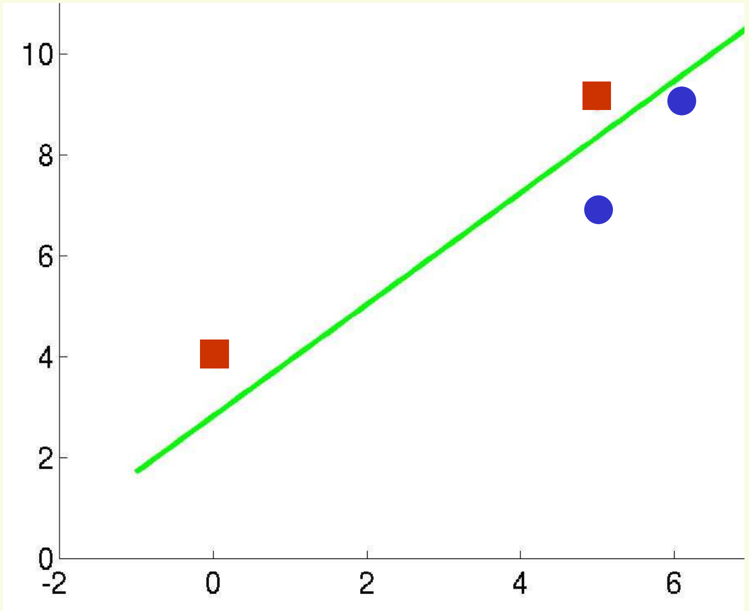
$$\boldsymbol{y_1} = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix} \quad \boldsymbol{y_2} = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} \quad \boldsymbol{y_3} = \begin{bmatrix} -1 \\ -5 \\ -9 \end{bmatrix} \quad \boldsymbol{y_4} = \begin{bmatrix} -1 \\ 0 \\ -4 \end{bmatrix}$$

- Matrix $\boldsymbol{Y}$ is then

$$\boldsymbol{Y} = \begin{bmatrix} 1 & 6 & 9 \\ 1 & 5 & 7 \\ -1 & -5 & -9 \\ -1 & 0 & -4 \end{bmatrix}$$

LDF: Example

- Choose $\boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$



- In matlab, ***a***=***Y\b*** solves the least squares problem

$$\boldsymbol{a} = \begin{bmatrix} 2.7 \\ 1.0 \\ -0.9 \end{bmatrix}$$

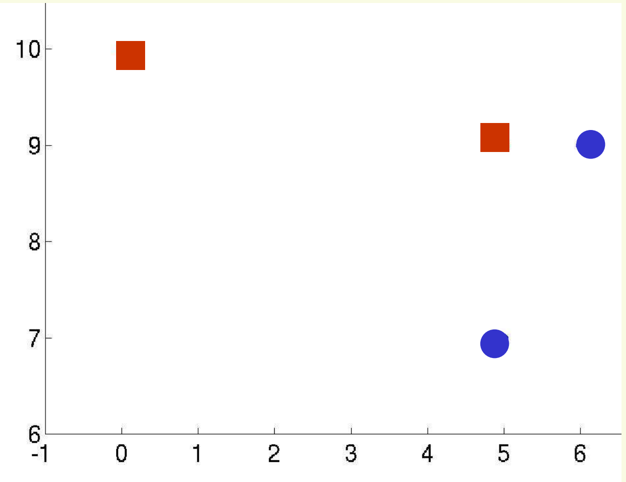
- Note ***a*** is an approximation to ***Ya*** = ***b***, since no exact solution exists

$$\boldsymbol{Ya} = \begin{bmatrix} 0.4 \\ 1.3 \\ 0.6 \\ 1.1 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- This solution does give a separating hyperplane since ***Ya*** > ***0***

LDF: Example

- Class 1: (6 9), (5 7)
- Class 2: (5 9), (0 10)
- The last sample is very far compared to others from the separating hyperplane

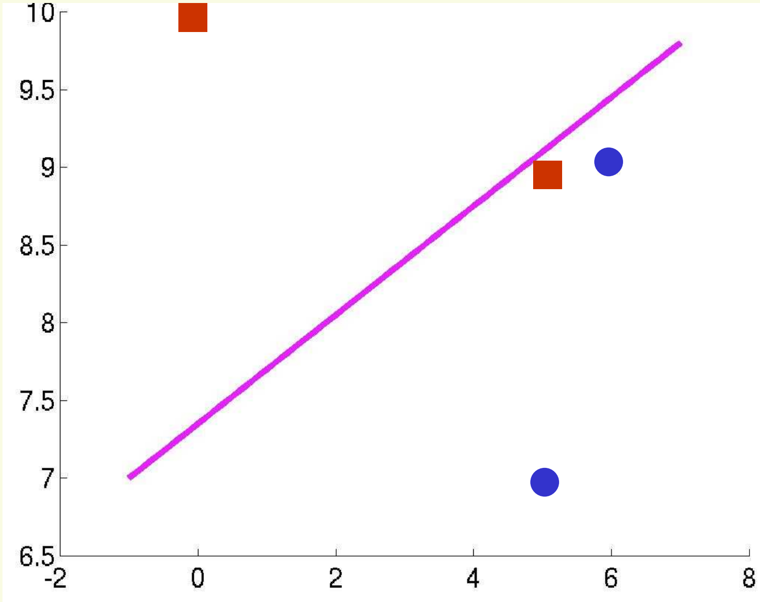


$$y_1 = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} \quad y_3 = \begin{bmatrix} -1 \\ -5 \\ -9 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ 0 \\ -10 \end{bmatrix}$$

- Matrix $Y = \begin{bmatrix} 1 & 6 & 9 \\ 1 & 5 & 7 \\ -1 & -5 & -9 \\ -1 & 0 & -10 \end{bmatrix}$

LDF: Example

- Choose $\boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$



- In matlab, ***a***=***Y******b*** solves the least squares problem

$$\boldsymbol{a} = \begin{bmatrix} 3.2 \\ 0.2 \\ -0.4 \end{bmatrix}$$

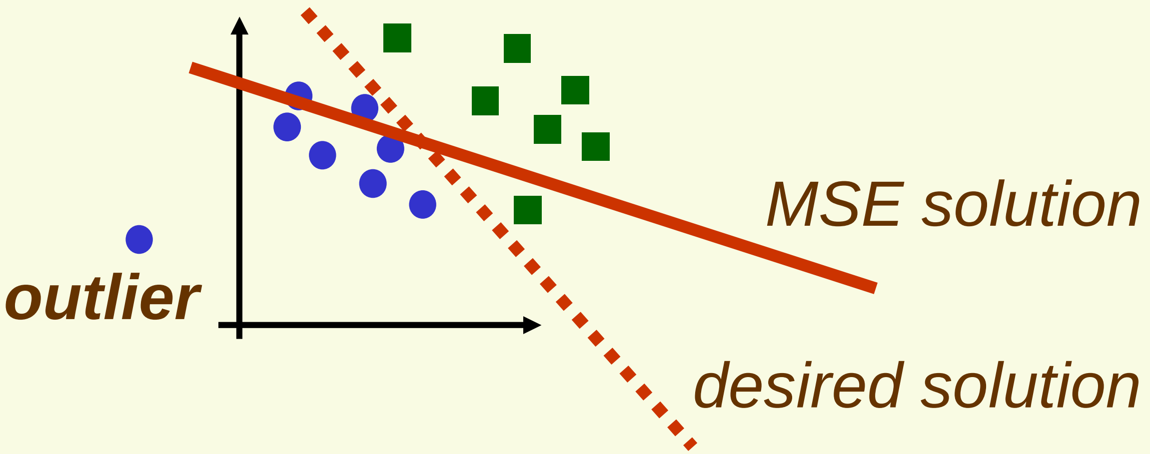
- Note ***a*** is an approximation to ***Ya*** = ***b***, since no exact solution exists

$$\boldsymbol{Ya} = \begin{bmatrix} 0.2 \\ 0.9 \\ -0.04 \\ 1.16 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- This solution does not give a separating hyperplane since $\boldsymbol{a}^t \boldsymbol{y}_3 < 0$

LDF: Example

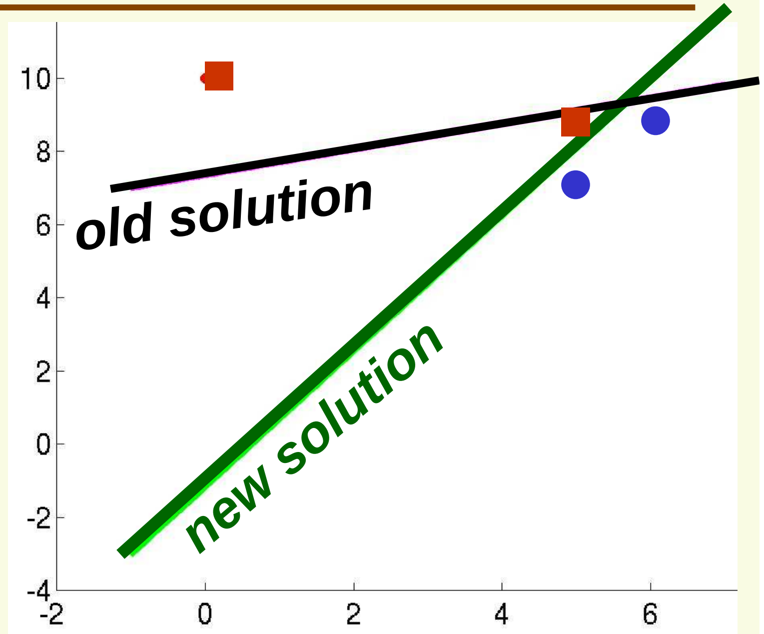
- MSE pays to much attention to isolated “noisy” examples (such examples are called outliers)



- No problems with convergence though, and solution it gives ranges from reasonable to good

LDF: Example

- we know that 4th point is far far from separating hyperplane
  - In practice we don't know this
- Thus appropriate $\boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 10 \end{bmatrix}$



- In Matlab, solve ***a=Y\b***

$$\boldsymbol{a} = \begin{bmatrix} -1.1 \\ 1.7 \\ -0.9 \end{bmatrix}$$

- Note ***a*** is an approximation to ***Ya = b***, $\boldsymbol{Ya} = \begin{bmatrix} 0.9 \\ 1.0 \\ 0.8 \\ 10.0 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 10 \end{bmatrix}$
- This solution does give the separating hyperplane since ***Ya*** > ***0***

LDF: Gradient Descent for MSE solution

$$J_s(\mathbf{a}) = \|\mathbf{Ya} - \mathbf{b}\|^2$$

- May wish to find MSE solution by gradient descent:
  1. Computing the inverse of $\mathbf{Y}^t\mathbf{Y}$ may be too costly
  2. $\mathbf{Y}^t\mathbf{Y}$ may be close to singular if samples are highly correlated (rows of $\mathbf{Y}$ are almost linear combinations of each other)
     - computing the inverse of $\mathbf{Y}^t\mathbf{Y}$ is not numerically stable

- In the beginning of the lecture, computed the gradient:

$$\nabla J_s(\mathbf{a}) = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b})$$

LDF: Widrow-Hoff Procedure

$$\nabla J_s(\boldsymbol{a}) = 2\boldsymbol{Y}^t(\boldsymbol{Ya} - \boldsymbol{b})$$

- Thus the update rule for gradient descent:

$$\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} - \eta^{(k)}\boldsymbol{Y}^t\left(\boldsymbol{Ya}^{(k)} - \boldsymbol{b}\right)$$

  - If $\eta^{(k)} = \eta^{(1)}/k$ weight vector $\boldsymbol{a}^{(k)}$ converges to the MSE solution $\boldsymbol{a}$, that is $\boldsymbol{Y}^t(\boldsymbol{Ya}-\boldsymbol{b})=0$

- *Widrow-Hoff procedure* reduces storage requirements by considering single samples sequentially:

$$\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} - \eta^{(k)}\boldsymbol{y}_i\left(\boldsymbol{y}_i^t\boldsymbol{a}^{(k)} - \boldsymbol{b}_i\right)$$

LDF: Ho-Kashyap Procedure

- In the MSE procedure, if $\boldsymbol{b}$ is chosen arbitrarily, finding separating hyperplane is not guaranteed
- Suppose training samples are linearly separable. Then there is $\boldsymbol{a}^s$ and positive $\boldsymbol{b}^s$ s.t.

$$\boldsymbol{Y}\boldsymbol{a}^s = \boldsymbol{b}^s > \mathbf{0}$$

- If we knew $\boldsymbol{b}^s$ could apply MSE procedure to find the separating hyperplane
- Idea: find both $\boldsymbol{a}^s$ and $\boldsymbol{b}^s$
- Minimize the following criterion function, restricting to positive $\boldsymbol{b}$:

$$J_{HK}(\boldsymbol{a},\boldsymbol{b}) = \|\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{b}\|^2$$

- $J_{HK}(\boldsymbol{a}^s,\boldsymbol{b}^s)=0$

LDF: Ho-Kashyap Procedure

$$J_{HK}(\mathbf{a},\mathbf{b}) = \|Y\mathbf{a}-\mathbf{b}\|^2$$

- As usual, take partial derivatives w.r.t. ***a*** and ***b***

$$\nabla_a J_{HK} = 2Y^t(Y\mathbf{a}-\mathbf{b}) = \mathbf{0}$$
$$\nabla_b J_{HK} = -2(Y\mathbf{a}-\mathbf{b}) = \mathbf{0}$$

- Use modified gradient descent procedure to find a minimum of $J_{HK}(\mathbf{a},\mathbf{b})$
- Alternate the two steps below until convergence:
  1) Fix ***b*** and minimize $J_{HK}(\mathbf{a},\mathbf{b})$ with respect to ***a***
  2) Fix ***a*** and minimize $J_{HK}(\mathbf{a},\mathbf{b})$ with respect to ***b***

LDF: Ho-Kashyap Procedure

$$\nabla_a J_{HK} = 2Y^t(Ya - b) = 0 \qquad \nabla_b J_{HK} = -2(Ya - b) = 0$$

- Alternate the two steps below until convergence:
  1) Fix ***b*** and minimize $J_{HK}(a,b)$ with respect to ***a***
  2) Fix ***a*** and minimize $J_{HK}(a,b)$ with respect to ***b***
- Step (1) can be performed with pseudoinverse
  - For fixed ***b*** minimum of $J_{HK}(a,b)$ with respect to ***a*** is found by solving

$$2Y^t(Ya - b) = 0$$

  - Thus

$$a = (Y^tY)^{-1}Y^tb$$

LDF: Ho-Kashyap Procedure

- Step 2: fix ***a*** and minimize $\boldsymbol{J_{HK}}(\boldsymbol{a},\boldsymbol{b})$ with respect to ***b***
- We can't use ***b*** = ***Ya*** because ***b*** has to be positive
- Solution: use modified gradient descent
- Regular gradient descent rule:

$$\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)} - \eta^{(k)} \nabla_{\boldsymbol{b}} \boldsymbol{J}\left(\boldsymbol{a}^{(k)}, \boldsymbol{b}^{(k)}\right)$$

- If any components of $\nabla_{\boldsymbol{b}} \boldsymbol{J}$ are positive, ***b*** will decrease and can possibly become negative

$$\boldsymbol{b}^{(k+1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - 2 * \begin{bmatrix} 2 \\ -3 \\ -2 \end{bmatrix} = \begin{bmatrix} -3 \\ 7 \\ 5 \end{bmatrix}$$

LDF: Ho-Kashyap Procedure

- start with positive ***b*** , follow negative gradient but refuse to decrease any components of ***b***

- This can be achieved by setting all the positive components of $\nabla_b J$ to ***0***

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} - \eta \frac{1}{2}\left[\nabla_b J\left(\mathbf{a}^{(k)}, \mathbf{b}^{(k)}\right) - |\nabla_b J\left(\mathbf{a}^{(k)}, \mathbf{b}^{(k)}\right)|\right]$$

  - here |***v***| denotes vector we get after applying absolute value to all elements of ***v***

$$\mathbf{b}^{(k+1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - 2 * \frac{1}{2}\left[\begin{bmatrix} 2 \\ -3 \\ -2 \end{bmatrix} - \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix}\right] = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -6 \\ -4 \end{bmatrix} = \begin{bmatrix} 1 \\ 7 \\ 5 \end{bmatrix}$$

- Not doing steepest descent anymore, but we are still doing descent ***and*** ensure that ***b*** is positive

LDF: Ho-Kashyap Procedure

$$\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)} - \eta \frac{1}{2}\left[\nabla_b J\left(\boldsymbol{a}^{(k)}, \boldsymbol{b}^{(k)}\right) - |\nabla_b J\left(\boldsymbol{a}^{(k)}, \boldsymbol{b}^{(k)}\right)|\right]$$

$$\nabla_b J = -2(Y\boldsymbol{a} - \boldsymbol{b}) = 0$$

- Let $\boldsymbol{e}^{(k)} = Y\boldsymbol{a}^{(k)} - \boldsymbol{b}^{(k)} = -\frac{1}{2}\nabla J_b\left(\boldsymbol{a}^{(k)}, \boldsymbol{b}^{(k)}\right)$

- Then

$$\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)} - \eta \frac{1}{2}\left[-2\boldsymbol{e}^{(k)} - |2\boldsymbol{e}^{(k)}|\right]$$

$$= \boldsymbol{b}^{(k)} + \eta\left[\boldsymbol{e}^{(k)} + |\boldsymbol{e}^{(k)}|\right]$$

LDF: Ho-Kashyap Procedure

- The final Ho-Kashyap procedure:

0) Start with arbitrary $\boldsymbol{a}^{(1)}$ and $\boldsymbol{b}^{(1)} > 0$, let k = 1

repeat steps (1) through (4)

1) $\boldsymbol{e}^{(k)} = \boldsymbol{Y}\boldsymbol{a}^{(k)} - \boldsymbol{b}^{(k)}$

2) Solve for $\boldsymbol{b}^{(k+1)}$ using $\boldsymbol{a}^{(k)}$ and $\boldsymbol{b}^{(k)}$

$$\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)} + \eta\left[\boldsymbol{e}^{(k)} + |\boldsymbol{e}^{(k)}|\right]$$

3) Solve for $\boldsymbol{a}^{(k+1)}$ using $\boldsymbol{b}^{(k+1)}$

$$\boldsymbol{a}^{(k+1)} = \left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^t\,\boldsymbol{b}^{(k+1)}$$

4) k = k + 1

until $\boldsymbol{e}^{(k)} >= 0$ or $\boldsymbol{k} > \boldsymbol{k}_{max}$ or $\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)}$

- For convergence, learning rate should be fixed between $0 < \eta < 1$

LDF: Ho-Kashyap Procedure

$$\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)} + \eta\left[\boldsymbol{e}^{(k)} + |\,\boldsymbol{e}^{(k)}\,|\right]$$

- What if $\boldsymbol{e}^{(k)}$ is negative for all components?
  - $\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)}$ and corrections stop
- Write $\boldsymbol{e}^{(k)}$ out:

$$\boldsymbol{e}^{(k)} = \boldsymbol{Ya}^{(k)} - \boldsymbol{b}^{(k)} = \boldsymbol{Y}\left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^t\boldsymbol{b}^{(k)} - \boldsymbol{b}^{(k)}$$

- Multiply by $\boldsymbol{Y}^t$:

$$\boldsymbol{Y}^t\boldsymbol{e}^{(k)} = \boldsymbol{Y}^t\left(\boldsymbol{Y}\left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^t\boldsymbol{b}^{(k)} - \boldsymbol{b}^{(k)}\right) = \boldsymbol{Y}^t\boldsymbol{b}^{(k)} - \boldsymbol{Y}^t\boldsymbol{b}^{(k)} = \boldsymbol{0}$$

- Thus $\boldsymbol{Y}^t\boldsymbol{e}^{(k)} = \boldsymbol{0}$

LDF: Ho-Kashyap Procedure

- Thus $\boldsymbol{Y}^t \boldsymbol{e}^{(k)} = \boldsymbol{0}$
- Suppose training samples are linearly separable. Then there is $\boldsymbol{a}^s$ and positive $\boldsymbol{b}^s$ s.t.

$$\boldsymbol{Y}\boldsymbol{a}^s = \boldsymbol{b}^s > \mathbf{0}$$

- Multiply both sides by $(\boldsymbol{e}^{(k)})^t$

$$\mathbf{0} = \left(\boldsymbol{e}^{(k)}\right)^t \boldsymbol{Y}\boldsymbol{a}^s = \left(\boldsymbol{e}^{(k)}\right)^t \boldsymbol{b}^s$$

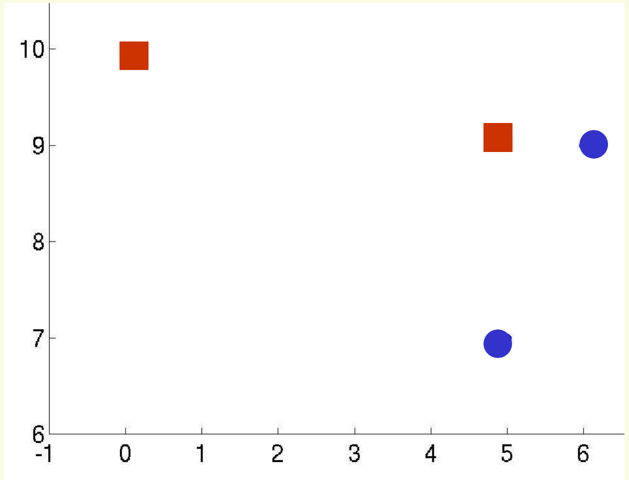
- Either $\boldsymbol{e}^{(k)} = \boldsymbol{0}$ or one of its components is positive

LDF: Ho-Kashyap Procedure

- In the linearly separable case,
  - $\boldsymbol{e}^{(k)} = \boldsymbol{0}$, found solution, stop
  - one of components of $\boldsymbol{e}^{(k)}$ is positive, algorithm continues

- In non separable case,
  - $\boldsymbol{e}^{(k)}$ will have only negative components eventually, thus found proof of nonseparability
  - No bound on how many iteration need for the proof of nonseparability

LDF: Ho-Kashyap Procedure Example



- Class 1: (6 9), (5 7)
- Class 1: (5 9), (0 10)
- Matrix $\boldsymbol{Y} = \begin{bmatrix} 1 & 6 & 9 \\ 1 & 5 & 7 \\ -1 & -5 & -9 \\ -1 & 0 & -10 \end{bmatrix}$
- Start with $\boldsymbol{a}^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $\boldsymbol{b}^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
- Use fixed learning $\eta = 0.9$
- At the start $\boldsymbol{Ya}^{(1)} = \begin{bmatrix} 16 \\ 13 \\ -15 \\ -11 \end{bmatrix}$

LDF: Ho-Kashyap Procedure Example

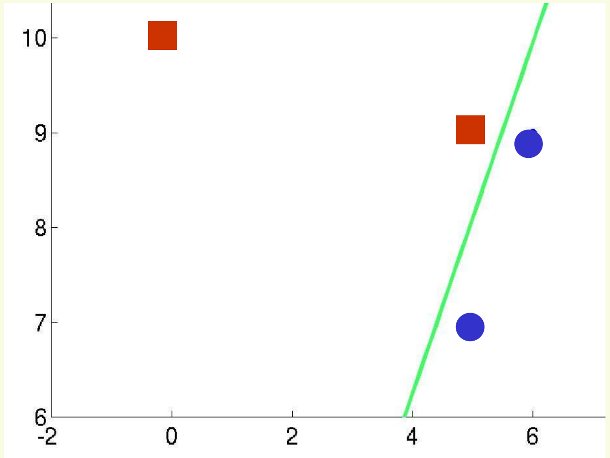
- Iteration 1:
  - $$e^{(1)} = Ya^{(1)} - b^{(1)} = \begin{bmatrix} 16 \\ 13 \\ -15 \\ -11 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 15 \\ 12 \\ -16 \\ -12 \end{bmatrix}$$
  - solve for $b^{(2)}$ using $a^{(1)}$ and $b^{(1)}$

$$b^{(2)} = b^{(1)} + 0.9\left[e^{(1)} + |e^{(1)}|\right] = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + 0.9\left[\begin{bmatrix} 15 \\ 12 \\ -16 \\ -12 \end{bmatrix} + \begin{bmatrix} 15 \\ 12 \\ 16 \\ 12 \end{bmatrix}\right] = \begin{bmatrix} 28 \\ 22.6 \\ 1 \\ 1 \end{bmatrix}$$

  - solve for $a^{(2)}$ using $b^{(2)}$

$$a^{(2)} = \left(Y^t Y\right)^{-1} Y^t b^{(2)} = \begin{bmatrix} -2.6 & 4.7 & 1.6 & -0.5 \\ 0.16 & -0.1 & -0.1 & 0.2 \\ 0.26 & -0.5 & -0.2 & -0.1 \end{bmatrix} * \begin{bmatrix} 28 \\ 22.6 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 34.6 \\ 2.7 \\ -3.8 \end{bmatrix}$$

LDF: Ho-Kashyap Procedure Example



- Continue iterations until $\mathbf{Ya} > \mathbf{0}$
  - In practice, continue until minimum component of $\mathbf{Ya}$ is less then 0.01
- After 104 iterations converged to solution

$$\mathbf{a} = \begin{bmatrix} -34.9 \\ 27.3 \\ -11.3 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 28 \\ 23 \\ 1 \\ 147 \end{bmatrix}$$

- $\mathbf{a}$ does gives a separating hyperplane

$$\mathbf{Ya} = \begin{bmatrix} 27.2 \\ 22.5 \\ 0.14 \\ 1.48 \end{bmatrix}$$

LDF: MSE for Multiple Classes

- Suppose we have m classes
- Define m linear discriminant functions

$$g_i(x) = w_i^t x + w_{i0} \qquad \mathrm{i = 1,...,m}$$

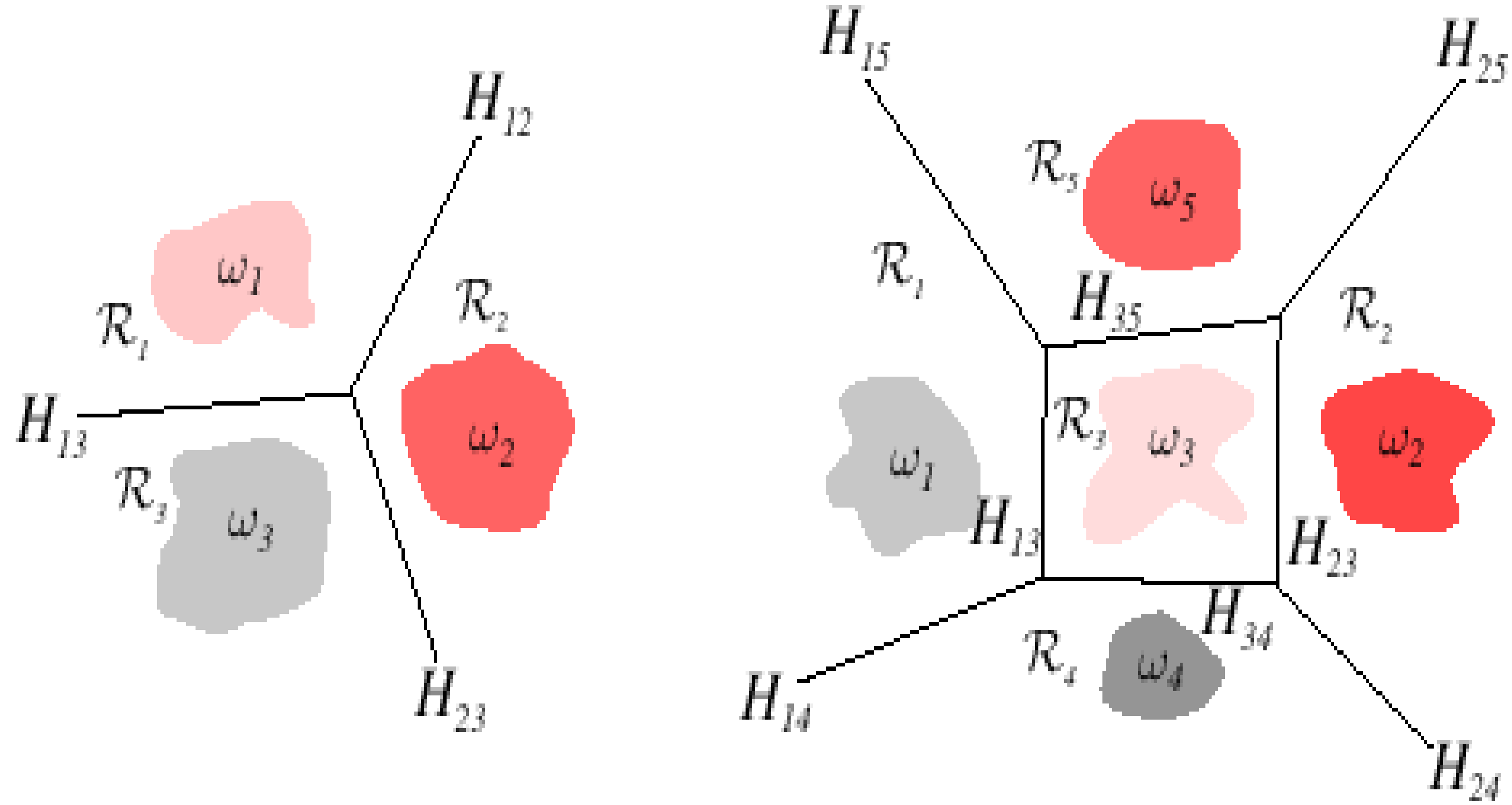
- Given x, assign class c_i if

$$g_i(x) \geq g_j(x) \qquad \forall \mathrm{j} \neq \mathrm{i}$$

- Such classifier is called a ***linear machine***

- A linear machine divides the feature space into c decision regions, with $g_i(x)$ being the largest discriminant if x is in the region R_i

LDF: Many Classes



LDF: MSE for Multiple Classes

- We still use augmented feature vectors $\boldsymbol{y_1},\ldots,\boldsymbol{y_n}$
- Define $\boldsymbol{m}$ linear discriminant functions

$$g_i(y) = a_i^t y \qquad \mathbf{i = 1,\ldots,m}$$

- Given $\boldsymbol{y}$, assign class $\boldsymbol{c_i}$ if

$$a_i^t y \geq a_j^t \mathbf{y} \qquad \forall \mathbf{j \neq i}$$

- For each class $\boldsymbol{i}$, makes sense to seek weight vector $\boldsymbol{a_i}$, s.t.

$$\begin{cases} a_i^t y = 1 & \forall \mathbf{y} \in \textbf{class i} \\ a_i^t y = 0 & \forall \mathbf{y} \notin \textbf{class i} \end{cases}$$

- If we find such $\boldsymbol{a_1},\ldots,\boldsymbol{a_m}$ the training error will be $\boldsymbol{0}$

LDF: MSE for Multiple Classes

- For each class ***i***, find weight vector $\boldsymbol{a_i}$, s.t.

$$\begin{cases} \boldsymbol{a}_i^t \boldsymbol{y} = 1 & \forall \mathbf{y} \in \textbf{class i} \\ \boldsymbol{a}_i^t \boldsymbol{y} = 0 & \forall \mathbf{y} \notin \textbf{class i} \end{cases}$$

- We can solve for each $\boldsymbol{a_i}$ independently
- Let $\boldsymbol{n_i}$ be the number of samples in class ***i***
- Let $\boldsymbol{Y_i}$ be matrix whose rows are samples from class ***i***, so it has ***d*** **+*1*** columns and $\boldsymbol{n_i}$ rows
- Let's pile all samples in ***n*** by ***d*** **+*1*** matrix ***Y***:

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{Y}_1 \\ \boldsymbol{Y}_2 \\ \vdots \\ \boldsymbol{Y}_m \end{bmatrix} = \begin{bmatrix} \textit{sample from class } 1 \\ \textit{sample from class } 1 \\ \vdots \\ \textit{sample from class } m \\ \textit{sample from class } m \end{bmatrix}$$

LDF: MSE for Multiple Classes

- Let $\boldsymbol{b_i}$ be a column vector of length $\boldsymbol{n}$ which is $\boldsymbol{0}$ everywhere except rows corresponding to samples from class $\boldsymbol{i}$, where it is $\boldsymbol{1}$:

$$\boldsymbol{b_i} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{rows corresponding to samples from class } i \\ \\ \\ \end{matrix}$$

- We need to solve: $\boldsymbol{Ya_i} = \boldsymbol{b_i}$

$$\begin{bmatrix} \text{sample from class 1} \\ \text{sample from class 1} \\ \vdots \\ \text{sample from class } m \\ \text{sample from class } m \end{bmatrix} \begin{bmatrix} \text{weights } a_i \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

LDF: MSE for Multiple Classes

- We need to solve $\boldsymbol{Ya_i} = \boldsymbol{b_i}$
- Usually no exact solution since $\boldsymbol{Y}$ is overdetermined
- Use least squares to minimize norm of the error vector $\| \boldsymbol{Ya_i} - \boldsymbol{b_i} \|$
- LSE solution with pseudoinverse:

$$\boldsymbol{a_i} = \left(\boldsymbol{Y}^t \boldsymbol{Y}\right)^{-1} \boldsymbol{Y}^t \boldsymbol{b_i}$$

- Thus we need to solve ***m*** LSE problems, one for each class
- Can write these ***m*** LSE problems in one matrix

LDF: MSE for Multiple Classes

- Let's pile all $\boldsymbol{b_i}$ as columns in $\boldsymbol{n}$ by $\boldsymbol{c}$ matrix $\boldsymbol{B}$

$$\boldsymbol{B} = \left[\boldsymbol{b_1} \ \cdots \ \boldsymbol{b_n}\right]$$

- Let's pile all $\boldsymbol{a_i}$ as columns in $\boldsymbol{d} + \boldsymbol{1}$ by $\boldsymbol{m}$ matrix $\boldsymbol{A}$

$$\boldsymbol{A} = \left[\boldsymbol{a}_1 \ \cdots \ \boldsymbol{a}_m\right] = \left[\text{weights } a_1 \ \ \text{weights } a_2 \ \ \cdots \ \ \text{weights } a_m\right]$$

- $\boldsymbol{m}$ LSE problems can be represented in $\boldsymbol{YA} = \boldsymbol{B}$:

$$\underbrace{\begin{bmatrix} \text{sample from class 1} \\ \text{sample from class 1} \\ \text{sample from class 2} \\ \text{sample from class 3} \\ \text{sample from class 3} \\ \text{sample from class 3} \end{bmatrix}}_{\boldsymbol{Y}} \underbrace{\left[\text{weights for c1} \ \ \text{weights for c2} \ \ \text{weights for c3}\right]}_{\boldsymbol{A}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{B}}$$

LDF: MSE for Multiple Classes

- Our objective function is:

$$J(A) = \sum_{i=1}^{m} \left\| Ya_i - b_i \right\|^2$$

- $J(A)$ is minimized with the use of pseudoinverse

$$A = \left(Y^t Y\right)^{-1} YB$$

LDF: Summary

- ***Perceptron*** procedures
  - find a separating hyperplane in the linearly separable case,
  - do not converge in the non-separable case
  - can force convergence by using a decreasing learning rate, but are not guaranteed a reasonable stopping point
- ***MSE*** procedures
  - converge in separable and not separable case
  - may not find separating hyperplane if classes are linearly separable
  - use pseudoinverse if Y^tY is not singular and not too large
  - use gradient descent (Widrow-Hoff procedure) otherwise
- ***Ho-Kashyap*** procedures
  - always converge
  - find separating hyperplane in the linearly separable case
  - more costly