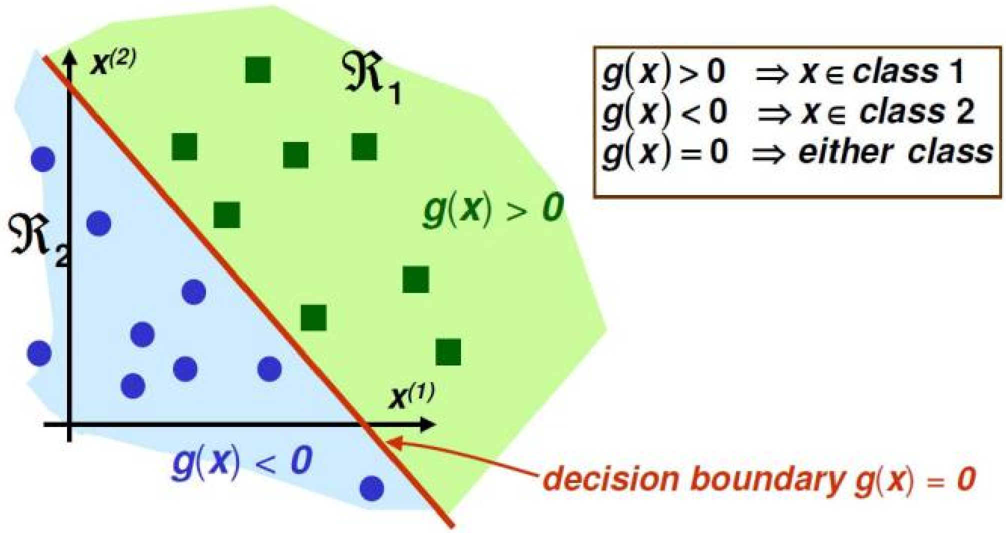
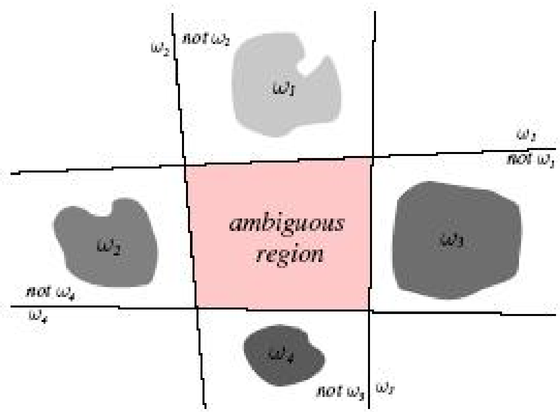
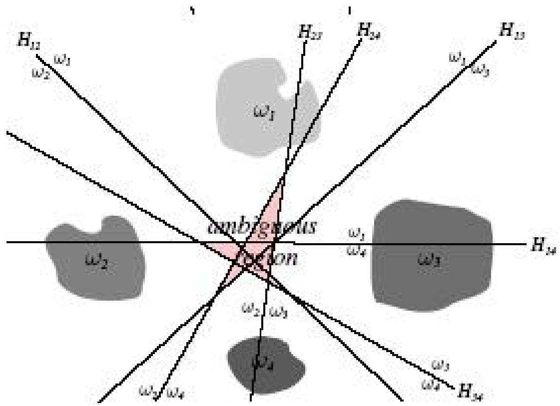
Lecture 10: Linear Discriminant Functions (2)

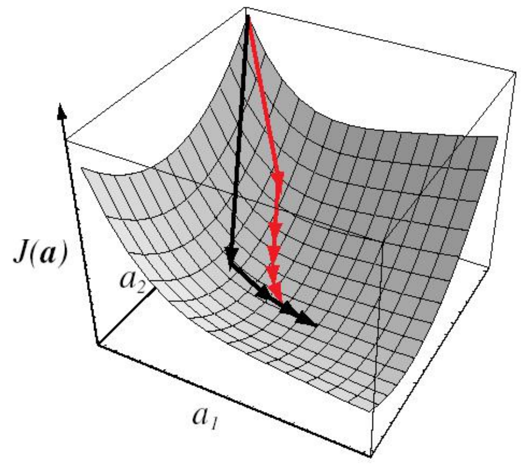
Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at RPI.
Email: **longc3@rpi.edu**

Recap Previous Lecture









Outline

- Perceptron Rule
- Minimum Squared-Error Procedure
- Ho–Kashyap Procedure

Outline

- **Perceptron Rule**
- Minimum Squared-Error Procedure
- Ho–Kashyap Procedure

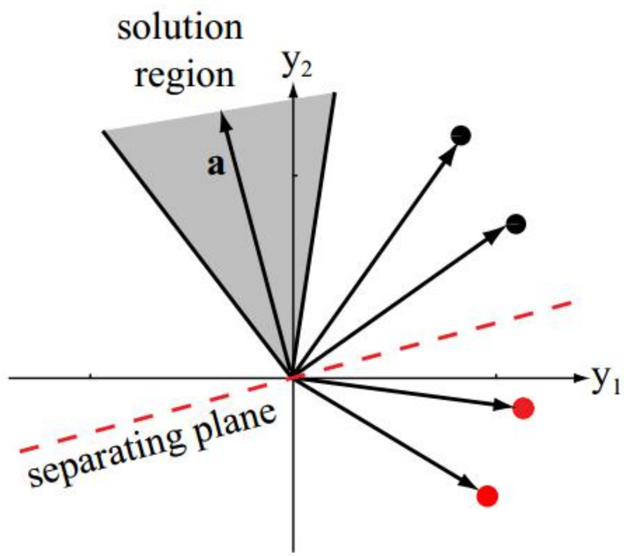
"Dual" Problem

Classification rule:

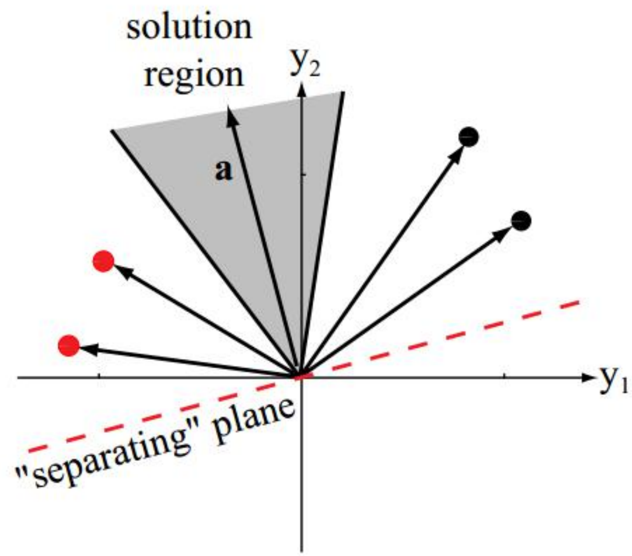
If $\boldsymbol{\alpha}^t\mathbf{y}_i>0$ assign $\mathbf{y}_i$ to ω_1

else if $\boldsymbol{\alpha}^t\mathbf{y}_i<0$ assign $\mathbf{y}_i$ to ω_2

- If $\mathbf{y}_i$ in ω_2, replace $\mathbf{y}_i$ by $-\mathbf{y}_i$
- Find $\boldsymbol{\alpha}$ such that: $\boldsymbol{\alpha}^t\mathbf{y}_i>0$



Seek a hyperplane that separates patterns from different categories



Seek a hyperplane that puts normalized patterns on the same (positive) side

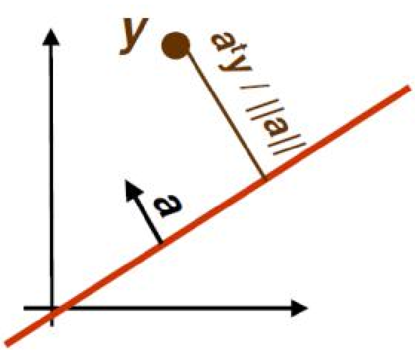
Perceptron rule

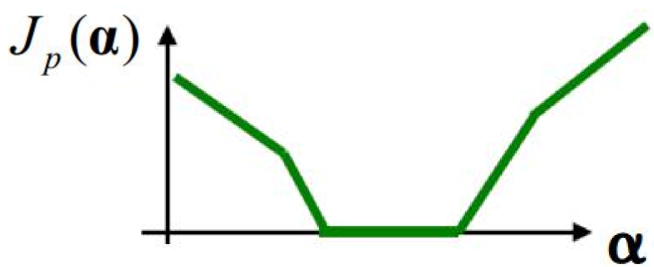
- Use Gradient Descent assuming that the error function to be minimized is:

$$J_p(\boldsymbol{\alpha}) = \sum_{\mathbf{y} \in Y(\boldsymbol{\alpha})} (-\boldsymbol{\alpha}^t \mathbf{y})$$

the set of samples misclassified by **α**.

- If $Y(\alpha)$ is empty, $J_p(\alpha)=0$; otherwise, $J_p(\alpha) \geq 0$.
- $J_p(\alpha)$ is $||\alpha||$ times the sum of distances of misclassified.
- $J_p(\alpha)$ is is piecewise linear and thus suitable for gradient descent.





Perceptron Batch Rule

- The gradient of $J_p(\boldsymbol{\alpha})$ is:

$$J_p(\boldsymbol{\alpha}) = \sum_{\mathbf{y} \in Y(\boldsymbol{\alpha})} (-\boldsymbol{\alpha}^t \mathbf{y}) \quad \Rightarrow \quad \nabla J_p = \sum_{\mathbf{y} \in Y(\boldsymbol{\alpha})} (-\mathbf{y})$$

- It is not possible to solve analytically $\nabla J_p = 0$.

- The perceptron update rule is obtained using gradient descent:

$$\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) + \eta(k) \sum_{\mathbf{y} \in Y(\boldsymbol{\alpha})} \mathbf{y}$$

- It is called batch rule because it is based on all misclassified examples

Perceptron Single Sample Rule

- The gradient decent single sample rule for $J_p(a)$ is:

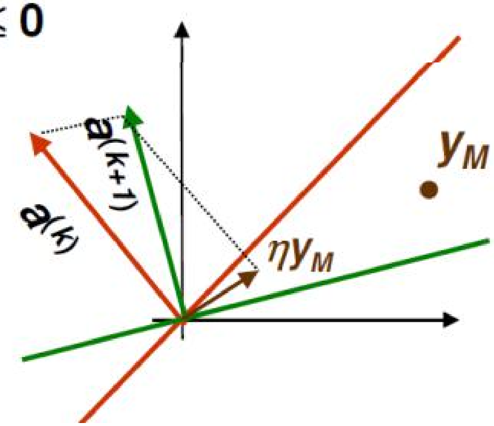
$$a^{(k+1)} = a^{(k)} + \eta^{(k)} y_M$$

  – Note that y_M is one sample misclassified by $a^{(k)}$
  – Must have a consistent way of visiting samples

- Geometric Interpretation:

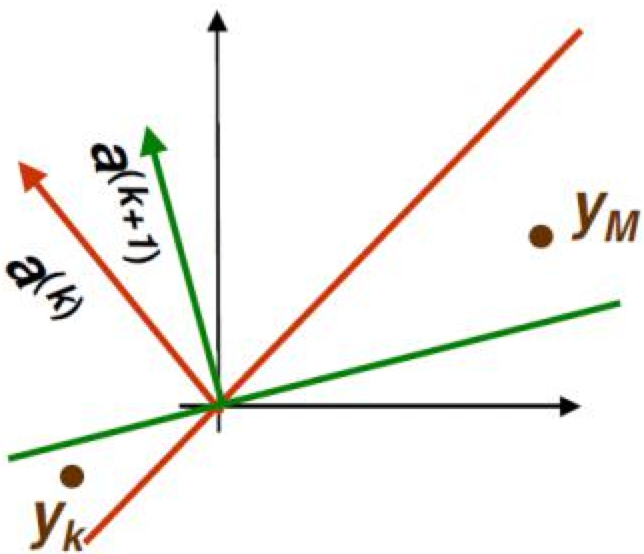
  – Note that y_M is one sample misclassified by $\left(a^{(k)}\right)^t y_M \leq 0$
  – yM is on the wrong side of decision hyperplane
  – Adding ηy_M to a moves the new decision hyperplane in the right direction with respect to y_M

$$\left(a^{(k)}\right)^t y_M \leq 0$$

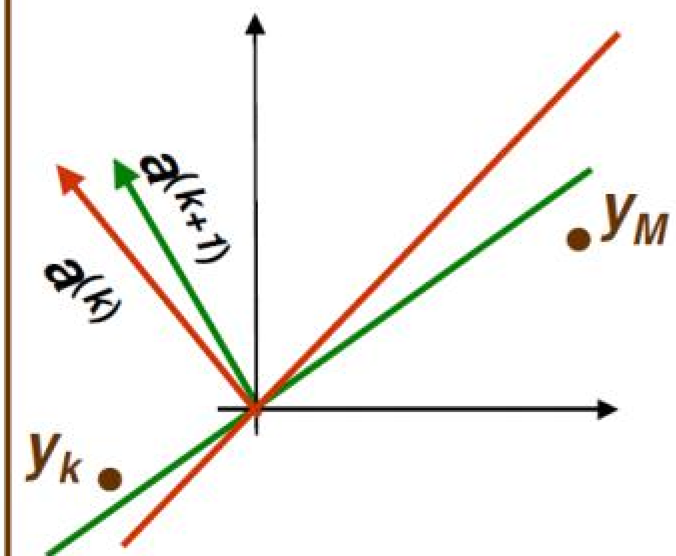


Perceptron Single Sample Rule

$$a^{(k+1)} = a^{(k)} + \eta^{(k)} y_M$$



η is too large, previously correctly classified sample y_k is now misclassified



η is too small, y_M is still misclassified

Perceptron Example

	features				grade
name	***good attendance?***	***tall?***	***sleeps in class?***	***chews gum?***	
Jane	*yes (1)*	*yes (1)*	*no (-1)*	*no (-1)*	*A*
Steve	*yes (1)*	*yes (1)*	*yes (1)*	*yes (1)*	*F*
Mary	*no (-1)*	*no (-1)*	*no (-1)*	*yes (1)*	*F*
Peter	*yes (1)*	*no (-1)*	*no (-1)*	*yes (1)*	*A*

- Class 1: students who get A
- Class 2: students who get F

Perceptron Example

	features				grade
name	***good attendance?***	***tall?***	***sleeps in class?***	***chews gum?***	
Jane	*yes (1)*	*yes (1)*	*no (-1)*	*no (-1)*	*A*
Steve	*yes (1)*	*yes (1)*	*yes (1)*	*yes (1)*	*F*
Mary	*no (-1)*	*no (-1)*	*no (-1)*	*yes (1)*	*F*
Peter	*yes (1)*	*no (-1)*	*no (-1)*	*yes (1)*	*A*

- **Augment samples** by adding an extra feature (dimension) equal to 1

Perceptron Example

	features				grade
name	***good attendance?***	***tall?***	***sleeps in class?***	***chews gum?***	
Jane	*yes (1)*	*yes (1)*	*no (-1)*	*no (-1)*	*A*
Steve	*yes (1)*	*yes (1)*	*yes (1)*	*yes (1)*	*F*
Mary	*no (-1)*	*no (-1)*	*no (-1)*	*yes (1)*	*F*
Peter	*yes (1)*	*no (-1)*	*no (-1)*	*yes (1)*	*A*

- **Normalize**:
  - Replace all examples from class 2 by their negative values $\quad \mathbf{y}_i \rightarrow -\mathbf{y}_i \quad \forall \mathbf{y}_i \in \mathbf{c}_2$
  - Seek ***a*** such that: $\quad \mathbf{a}^t \mathbf{y}_i > 0 \quad \forall \mathbf{y}_i$

Perceptron Example

	features				grade
name	***good attendance?***	***tall?***	***sleeps in class?***	***chews gum?***	
Jane	*yes (1)*	*yes (1)*	*no (-1)*	*no (-1)*	*A*
Steve	*yes (1)*	*yes (1)*	*yes (1)*	*yes (1)*	*F*
Mary	*no (-1)*	*no (-1)*	*no (-1)*	*yes (1)*	*F*
Peter	*yes (1)*	*no (-1)*	*no (-1)*	*yes (1)*	*A*

- **Single Sample Rule**:
  - Sample is misclassified if $a^t y_i = \sum_{k=0}^{4} a_k y_i^{(k)} < 0$
  - Gradient descent single sample rule: $a^{(k+1)} = a^{(k)} + \eta^{(k)} \sum_{y \in Y_M} y$
  - Set η fixed learning rate to $\eta^{(k)} = 1$: $\boxed{a^{(k+1)} = a^{(k)} + y_M}$

Perceptron Example

- Set equal initial weights

$$a^{(1)} = [0.25, 0.25, 0.25, 0.25, 0.25]$$

- Visit all samples sequentially, modifying the weights after each misclassified example

name	$a^t y$	*misclassified?*
Jane	$0.25*1+0.25*1+0.25*1+0.25*(-1)+0.25*(-1)>0$	*no*
Steve	$0.25*(-1)+0.25*(-1)+0.25*(-1)+0.25*(-1)+0.25*(-1)<0$	*yes*

- New weights

$$\begin{aligned} a^{(2)} = a^{(1)} + y_M &= [0.25\ \ 0.25\ \ 0.25\ \ 0.25\ \ 0.25] + \\ &+ [-1\ \ -1\ \ -1\ \ -1\ \ -1] = \\ &= [-0.75\ \ -0.75\ \ -0.75\ \ -0.75\ \ -0.75] \end{aligned}$$

Perceptron Example

$$a^{(2)} = [-0.75 \;\; -0.75 \;\; -0.75 \;\; -0.75 \;\; -0.75]$$

name	$a^t y$	***misclassified?***
Mary	*-0.75*(-1)-0.75*1 -0.75 *1 -0.75 *1 -0.75*(-1) <0*	*yes*

- New weights

$$a^{(3)} = a^{(2)} + y_M = [-0.75 \;\; -0.75 \;\; -0.75 \;\; -0.75 \;\; -0.75] +$$
$$+[-1 \;\; 1 \;\; 1 \;\; 1 \;\; -1] =$$
$$= [-1.75 \;\; 0.25 \;\; 0.25 \;\; 0.25 \;\; -1.75]$$

Perceptron Example

$$a^{(3)} = [-1.75 \quad 0.25 \quad 0.25 \quad 0.25 \quad -1.75]$$

name	$a^t y$	*misclassified?*
Peter	*-1.75 *1 +0.25* 1+0.25* (-1) +0.25 *(-1)-1.75*1 <0*	*yes*

- New weights

$$a^{(4)} = a^{(3)} + y_M = [-1.75 \quad 0.25 \quad 0.25 \quad 0.25 \quad -1.75] +$$
$$+ [1 \quad 1 \quad -1 \quad -1 \quad 1] =$$
$$= [-0.75 \quad 1.25 \quad -0.75 \quad -0.75 \quad -0.75]$$

Perceptron Example

$$a^{(4)} = [-0.75 \ \ 1.25 \ \ -0.75 \ -0.75 \ -0.75]$$

name	$a^t y$	misclassified?
Jane	-0.75 *1 +1.25*1 -0.75*1 -0.75 *(-1) -0.75 *(-1)+0	no
Steve	-0.75*(-1)+1.25*(-1) -0.75*(-1) -0.75*(-1)-0.75*(-1)>0	no
Mary	-0.75 *(-1)+1.25*1-0.75*1 -0.75 *1 –0.75*(-1) >0	no
Peter	-0.75 *1+ 1.25*1-0.75* (-1)-0.75* (-1) -0.75 *1 >0	no

- Thus the discriminant function is:

$$g(y) = -0.75 * y^{(0)} + 1.25 * y^{(1)} - 0.75 * y^{(2)} - 0.75 * y^{(3)} - 0.75 * y^{(4)}$$

- Converting back to the original features x:

$$g(x) = 1.25 * x^{(1)} - 0.75 * x^{(2)} - 0.75 * x^{(3)} - 0.75 * x^{(4)} - 0.75$$

Perceptron Example

- Converting back to the original features x:

$$1.25 * x^{(1)} - 0.75 * x^{(2)} - 0.75 * x^{(3)} - 0.75 * x^{(4)} > 0.75 \Rightarrow grade\ A$$
$$1.25 * x^{(1)} - 0.75 * x^{(2)} - 0.75 * x^{(3)} - 0.75 * x^{(4)} < 0.75 \Rightarrow grade\ F$$

good attendance | tall | sleeps in class | chews gum

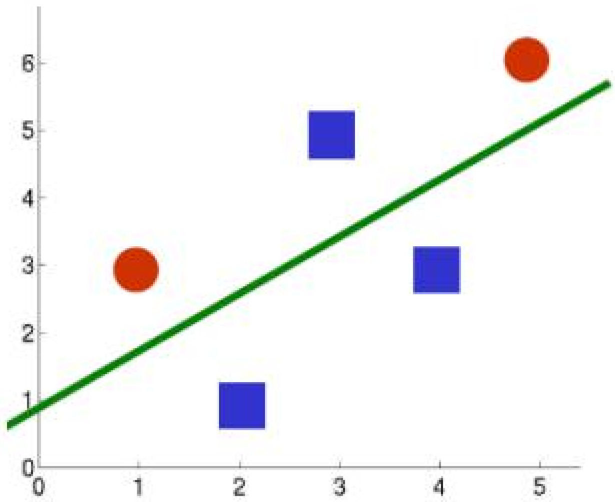
- This is just one possible solution vector.
- If we started with weights $a^{(1)}$=[0,0.5, 0.5, 0, 0] , the solution would be [-1,1.5, -0.5, -1, -1]

$$1.5 * x^{(1)} - 0.5 * x^{(2)} - x^{(3)} - x^{(4)} > 1 \Rightarrow grade\ A$$
$$1.5 * x^{(1)} - 0.5 * x^{(2)} - x^{(3)} - x^{(4)} < 1 \Rightarrow grade\ F$$

- In this solution, being tall is the least important feature

LDF: Non-separable Example

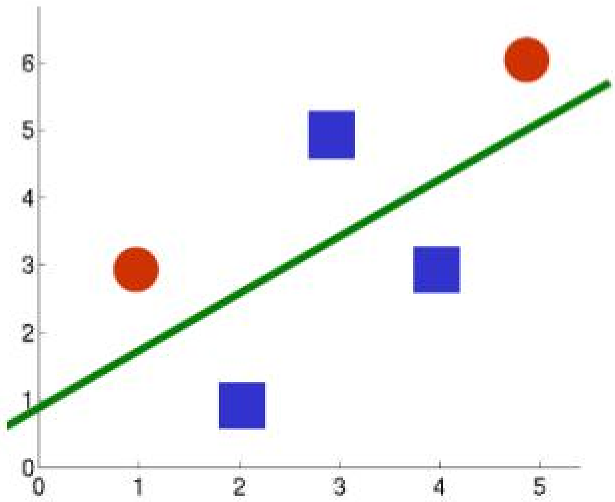
- Suppose we have 2 features and the samples are:
  - Class 1: $[2,1]$, $[4,3]$, $[3,5]$
  - Class 2: $[1,3]$ and $[5,6]$
- These samples are not separable by a line
- Still would like to get approximate separation by a line
  - A good choice is shown in green
  - Some samples may be "noisy", and we could accept them being misclassified



LDF: Non-separable Example

- Obtain y1 , y2 , y3 , y4 by adding extra feature and "normalizing"

$$y_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad y_3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad y_5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$



LDF: Non-separable Example

- Apply Perceptron single sample algorithm
- Initial equal weights

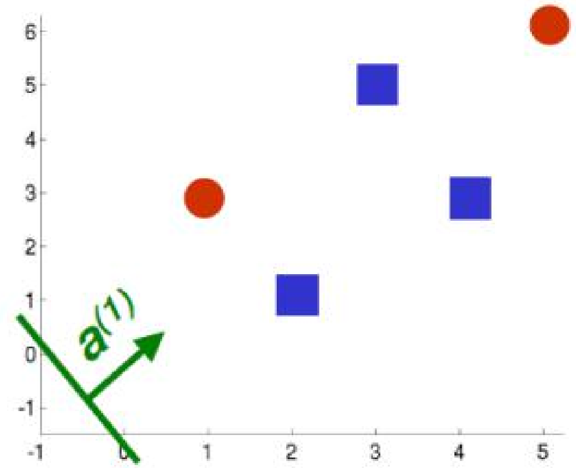
  $a^{(1)} = [1\ 1\ 1]$

  – Line equation $x^{(1)}+x^{(2)}+1=0$

- Fixed learning rate $\eta = 1$

$$a^{(k+1)} = a^{(k)} + y_M$$

$$y_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad y_3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad y_5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$

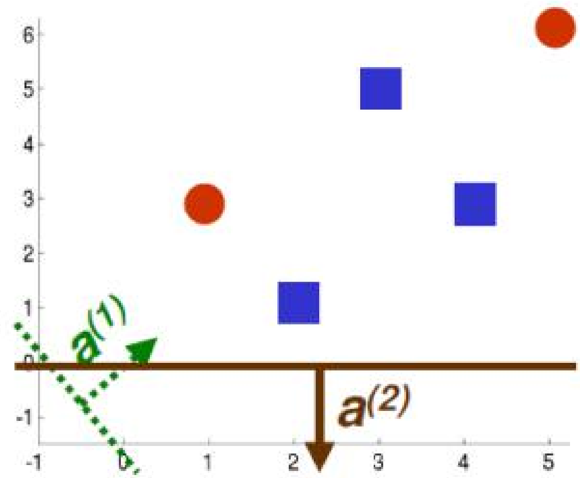


- $y^t_1 a^{(1)} = [1\ 1\ 1]^*[1\ 2\ 1]^t > 0$ ✔
- $y^t_2 a^{(1)} = [1\ 1\ 1]^*[1\ 4\ 3]^t > 0$ ✔
- $y^t_3 a^{(1)} = [1\ 1\ 1]^*[1\ 3\ 5]^t > 0$ ✔

LDF: Non-separable Example

$$a^{(1)} = [1 \ 1 \ 1] \qquad a^{(k+1)} = a^{(k)} + y_M$$

$$y_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad y_3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad y_5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$



- $y^t_4 a^{(1)}$=[1 1 1]*[-1 -1 -3]^t= -5< 0

$$a^{(2)} = a^{(1)} + y_M = [1 \ 1 \ 1] + [-1 \ -1 \ -3] = [0 \ 0 \ -2]$$

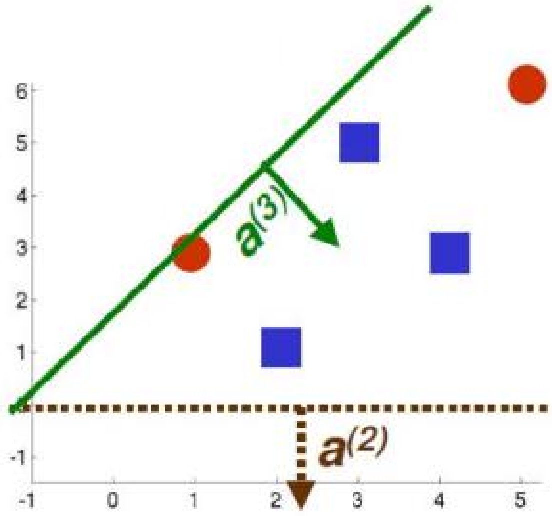
- $y^t_5 a^{(2)}$=[0 0 -2]*[-1 -5 -6]^t= 12 > 0 ✓
- $y^t_1 a^{(2)}$=[0 0 -2]*[1 2 1]^t < 0

$$a^{(3)} = a^{(2)} + y_M = [0 \ 0 \ -2] + [1 \ 2 \ 1] = [1 \ 2 \ -1]$$

LDF: Non-separable Example

$$a^{(3)} = [1\ 2\ -1] \qquad a^{(k+1)} = a^{(k)} + y_M$$

$$y_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad y_3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad y_5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$



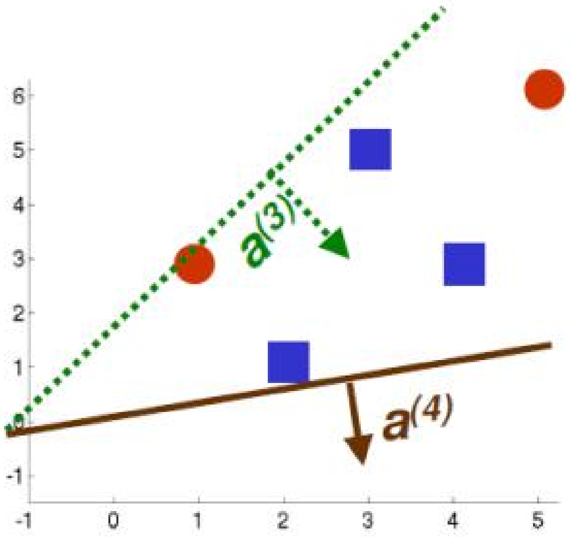
- $y^t{}_2\, a^{(3)}$=[1 4 3]*[1 2 -1]^t=6 > 0 ✔
- $y^t{}_3\, a^{(3)}$=[1 3 5]*[1 2 -1]^t > 0 ✔
- $y^t{}_4\, a^{(3)}$=[-1 -1 -3]*[1 2 -1]^t = 0

$$a^{(4)} = a^{(3)} + y_M = [1\ 2\ -1] + [-1\ -1\ -3] = [0\ 1\ -4]$$

LDF: Non-separable Example

$$a^{(4)} = [0\ 1\ -4] \qquad a^{(k+1)} = a^{(k)} + y_M$$

$$y_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad y_3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad y_5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$

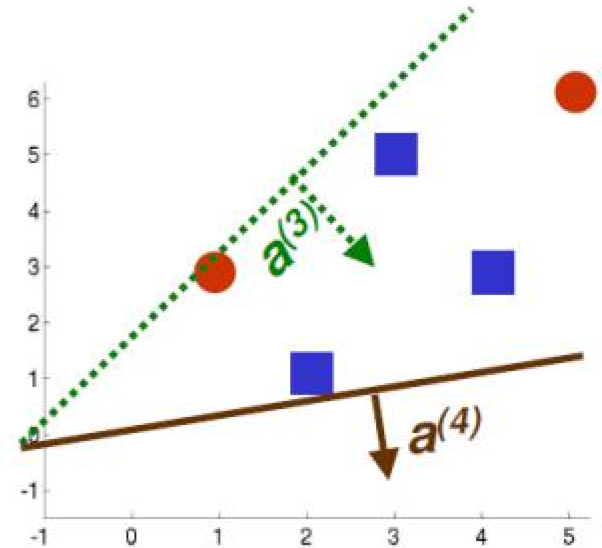


- $y_5^t a^{(4)}$=[-1 -5 -6]*[0 1 -4]=19>0
- $y_1^t a^{(4)}$=[1 2 1]*[0 1 -4]=-2<0
- ....

LDF: Non-separable Example

$$a^{(4)} = [0 \ 1 \ -4] \qquad a^{(k+1)} = a^{(k)} + y_M$$

$$y_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad y_3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad y_5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$

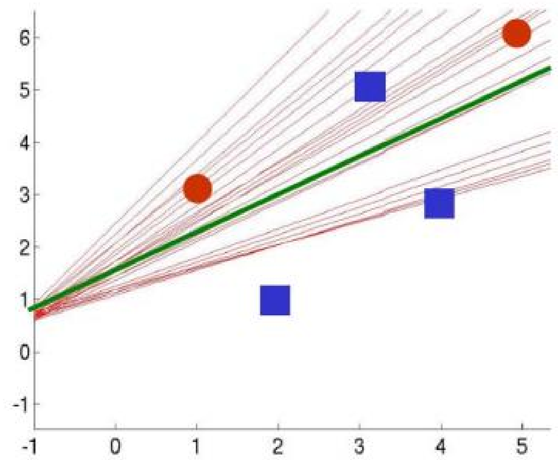


- $y_5^t a^{(4)}$=[-1 -5 -6]*[0 1 -4]=19>0
- $y_1^t a^{(4)}$=[1 2 1]*[0 1 -4]=-2<0
- ....

LDF: Non-separable Example

- We can continue this forever.
- There is no solution vector **a** satisfying for all **x_i**

$$a^t y_i = \sum_{k=0}^{5} a_k y_i^{(k)} > 0$$



- Need to stop but at a good point
- 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

Convergence of Perceptron Rules

- If classes are linearly separable and we use fixed learning rate, that is for $\eta(k)$ =const
- Then, both the single sample and batch perceptron rules converge to a correct solution (could be any a in the solution space)
- If classes are not linearly separable:
  – The algorithm does not stop, it keeps looking for a solution which does not exist
  – By choosing appropriate learning rate, we can always ensure convergence:
  – For example inverse linear learning rate:
  – For inverse linear learning rate, convergence in the linearly separable case can also be proven
  – No guarantee that we stopped at a good point, but there are good reasons to choose inverse linear learning rate

Perceptron Rule and Gradient decent

- Linearly separable data
  –perceptron rule with gradient decent works well
- Linearly non–separable data
  –need to stop perceptron rule algorithm at a good point, this maybe tricky

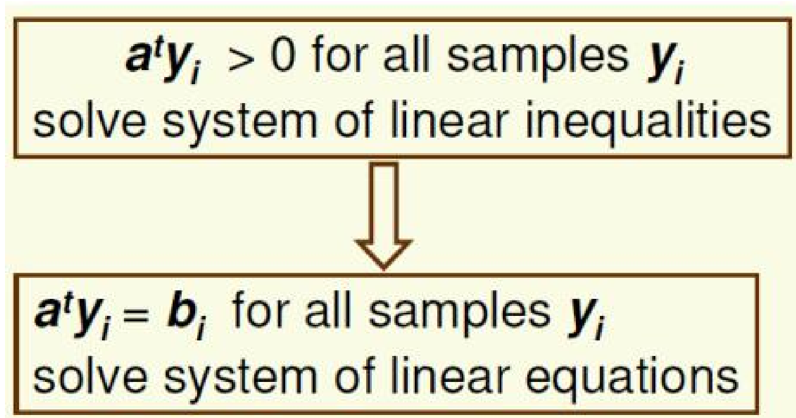
Batch Rule	*Single Sample Rule*
▪ Smoother gradient because all samples are used	▪ easier to analyze ▪ Concentrates more than necessary on any isolated “noisy” training examples

Outline

- Perceptron Rule
- **Minimum Squared-Error Procedure**
- Ho–Kashyap Procedure

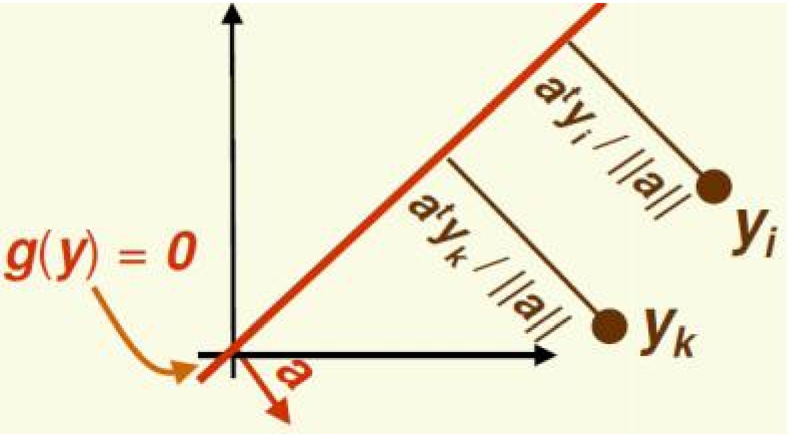
Minimum Squared-Error Procedures

- Idea: convert to easier and better understood problem



- MSE procedure
  - Choose positive constants $b_1, b_2, \ldots, b_n$
  - Try to find weight vector a such that $a^t y_i = b_i$ for all samples y_i
  - If we can find such a vector, then a is a solution because the bi's are positive
  - Consider all the samples (not just the misclassified ones)

MSE Margins



- If $a^t y_i = b_i$, y_i must be at distance b_i from the separating hyperplane (normalized by $||a||$)
- Thus $b_1, b_2, \ldots, b_n$ give relative expected distances or "margins" of samples from the hyperplane
- Should make b_i small if sample i is expected to be near separating hyperplane, and large otherwise
- In the absence of any additional information, set $b_1 = b_2 = \ldots = b_n = 1$

MSE Matrix Notation

- Need to solve n equations

$$\begin{cases} a^t y_1 = b_1 \\ \vdots \\ a^t y_n = b_n \end{cases}$$

- In matrix form Ya=b

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

Exact Solution is Rare

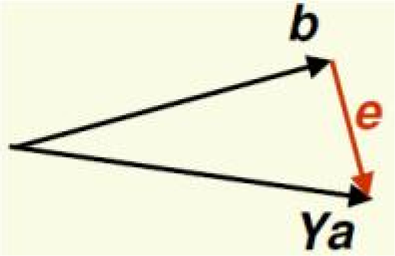
- Need to solve a linear system $Ya = b$
  - Y is an $n \times (d+1)$ matrix
- Exact solution only if Y is non−singular and square (the inverse Y^{-1} exists)
  - $a = Y^{-1}b$
  - (number of samples) = (number of features + 1)
  - Almost never happens in practice
  - Guaranteed to find the separating hyperplane

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 equations
  - More equations than unknowns
- Find an approximate solution
  - Note that approximate solution a does not necessarily give the separating hyperplane in the separable case
  - But the hyperplane corresponding to a may still be a good solution, especially if there is no separating hyperplane

MSE Criterion Function

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



$$\mathbf{e} = \mathbf{Y}\mathbf{a} - \mathbf{b}$$

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

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

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

Computing the Gradient

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

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

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

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

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

Pseudo-Inverse Solution

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

- Setting the gradient to 0:

$$2Y^t(Ya - b) = 0 \Rightarrow Y^tYa = Y^tb$$

- The matrix Y^tY is square $($it has d $+1$ rows and columns$)$ and it is often non−singular
- If Y^tY is non−singular, its inverse exists and we can solve for a uniquely:

$$a = \underbrace{\boxed{(Y^tY)^{-1}Y^t}}_{\text{pseudo inverse of } Y} b$$

$$((Y^tY)^{-1}Y^t)Y = (Y^tY)^{-1}(Y^tY) = I$$

MSE Procedures

- Only guaranteed separating hyperplane if $Ya \geq 0$
  - That is if all elements of vector Ya are positive

$$Ya = \begin{bmatrix} b_1 + \varepsilon_1 \\ \vdots \\ b_n + \varepsilon_n \end{bmatrix}$$

  - where ε may be negative

- If $\varepsilon_1, \ldots, \varepsilon_n$ are small relative to $b_1, \ldots, b_n$, then each element of Ya is positive, and a gives a separating hyperplane
  - If the approximation is not good, ε_i may be large and negative, for some i, thus $b_i + \varepsilon_i$ will be negative and a is not a separating hyperplane
- In linearly separable case, least squares solution a does not necessarily give separating hyperplane

MSE Procedures

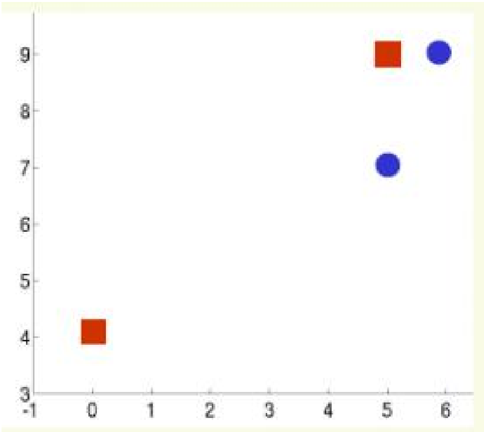
- We are free to choose b. We may be tempted to make b large as a way to ensure Ya $=$b > 0
  - Does not work
  - Let β be a scalar, let's try β b instead of b
- If a$*$ is a least squares solution to Ya $=$ b, then for any scalar β , the least squares solution to Ya $=$ β b is β a$*$

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

- Thus if the i–th element of Ya is less than 0, that is $y_i^t a < 0$, then $y_i^t(\beta a) < 0$
  - The relative difference between components of b matters, but not the size of each individual component

LDF using MSE: Example 1

- Class $1: (6\ \ 9), (5\ \ 7)$
- Class $2: (5\ \ 9), (0\ \ 4)$
- Add extra feature and "normalize"



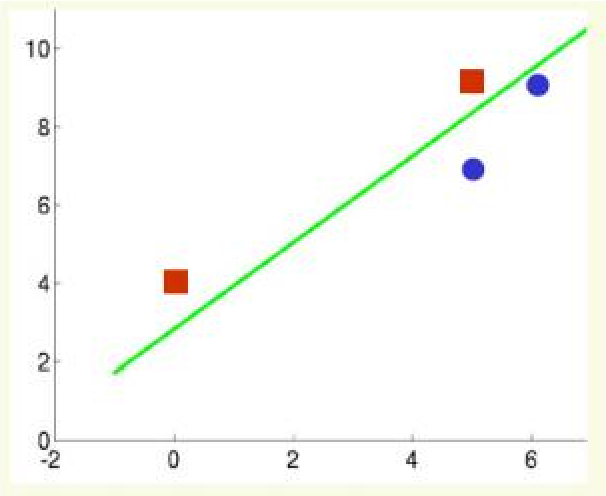
$$y_1 = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} \quad y_3 = \begin{bmatrix} -1 \\ -5 \\ -9 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ 0 \\ -4 \end{bmatrix}$$

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

LDF using MSE: Example 1

- Choose **b=[1 1 1 1]^T**
- In Matlab, **a=Y\b** solves the least squares problem

$$a = \begin{bmatrix} 2.66 \\ 1.045 \\ -0.944 \end{bmatrix}$$

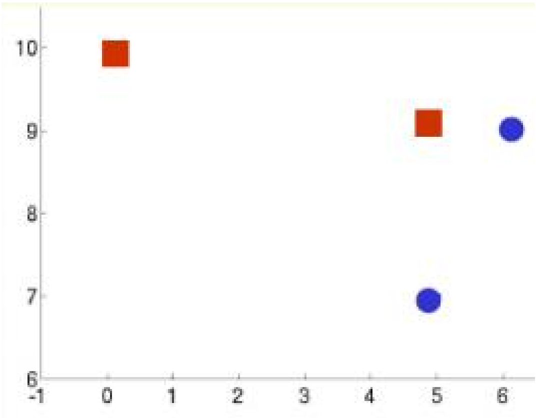


- Note a is an approximation to Ya $=$ b, since no exact solution exists
- This solution gives a separating hyperplane since Ya > 0

$$Ya = \begin{bmatrix} 0.44 \\ 1.28 \\ 0.61 \\ 1.11 \end{bmatrix}$$

LDF using MSE: Example 2

- 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



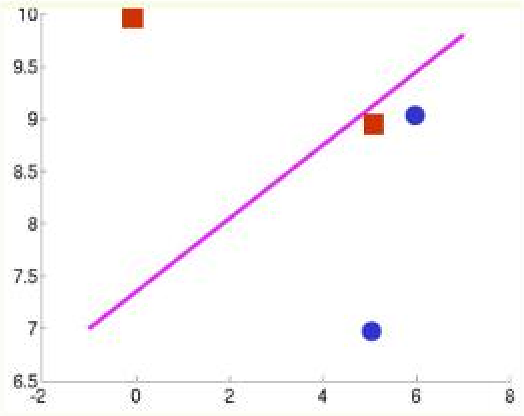
$$y_1 = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} \quad y_3 = \begin{bmatrix} -1 \\ -5 \\ -9 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ 0 \\ -10 \end{bmatrix}$$

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

LDF using MSE: Example 2

- Choose $b=[1\ 1\ 1\ 1]^T$
- In Matlab, **a=Y\b** solves the least squares problem

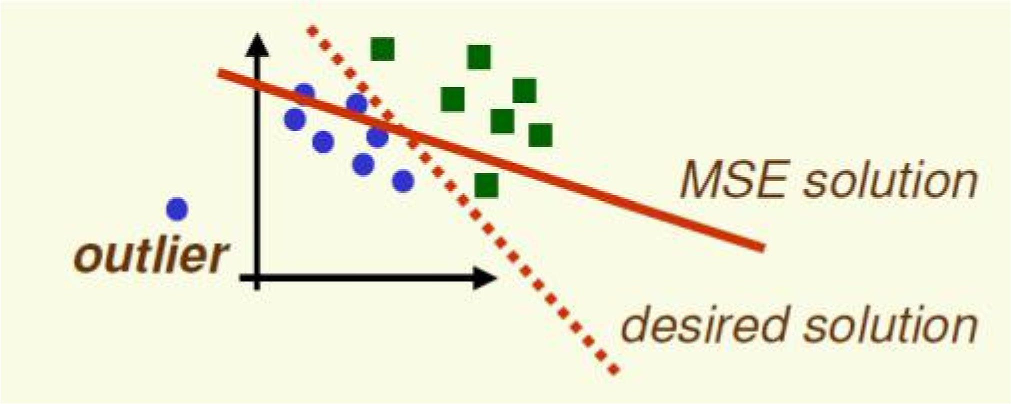
$$a = \begin{bmatrix} 3.2 \\ 0.2 \\ -0.4 \end{bmatrix} \qquad Ya = \begin{bmatrix} 0.2 \\ 0.9 \\ -0.04 \\ 1.16 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$



- This solution does not provide a separating hyperplane since $a^t y_3 < 0$

LDF using MSE: Example 2

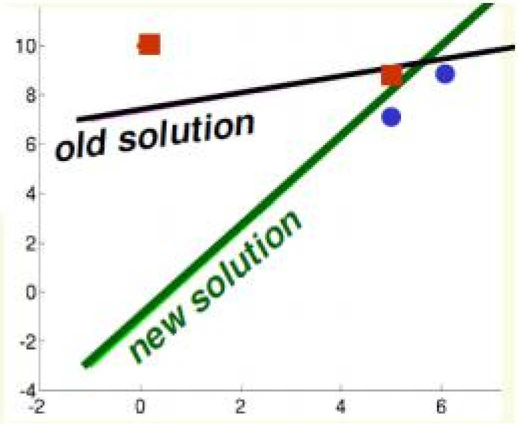
- MSE pays too much attention to isolated “noisy” examples
  – such examples are called outliers



- No problems with convergence
- Solution ranges from reasonable to good

LDF using MSE: Example 2

- We can see that the $4-$th point is vary far from separating hyperplane
  – In practice we don't know this
- A more appropriate b could be $b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 10 \end{bmatrix}$
- In Matlab, **a=Y\b** solves the least squares problem



$$a = \begin{bmatrix} -1.1 \\ 1.7 \\ -0.9 \end{bmatrix} \qquad Ya = \begin{bmatrix} 0.9 \\ 1.0 \\ 0.8 \\ 10.0 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 10 \end{bmatrix}$$

- This solution gives the separating hyperplane since $Ya > 0$

Gradient Descent for MSE

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

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

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

Widrow-Hoff Procedure

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

- Thus the update rule for gradient descent is:

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

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

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

$$a^{(k+1)} = a^{(k)} - \eta^{(k)} y_i \left(y_i^t a^{(k)} - b_i \right)$$

Outline

- Perceptron Rule
- Minimum Squared-Error Procedure
- **Ho–Kashyap Procedure**

Ho-Kashyap Procedure

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

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

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

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

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

Ho-Kashyap Procedure

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

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

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

- Use modified gradient descent procedure to find a minimum of $J_{HK}(a,b)$

- 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

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 $JHK(a,b)$ with respect to a is found by solving

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

  – Thus

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

Ho-Kashyap Procedure

- Step 2: fix a and minimize $J_{HK}(a,b)$ 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:

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

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

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

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 $\nabla_{\mathbf{b}}\mathbf{J}$ to **0**

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

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

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

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

Ho-Kashyap Procedure

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

$$\nabla_b J = -2(Ya - b) = 0$$

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

Then

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

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

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 $0 < \eta < 1$.

Ho-Kashyap Procedure

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

- What if $\mathbf{e}^{(k)}$ is negative for all components？

  $\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)}$ and corrections stop

- Write $\mathbf{e}^{(k)}$ out:

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

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

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

- Thus

$$\mathbf{Y}^t\mathbf{e}^{(k)} = 0$$

Ho-Kashyap Procedure

- Suppose training samples are linearly separable. Then there is $\mathbf{a}^s$ and positive $\mathbf{b}^s$ s.t

$$Ya^s = b^s > 0$$

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

$$0 = \left(e^{(k)}\right)^t Ya^s = \left(e^{(k)}\right)^t b^s$$

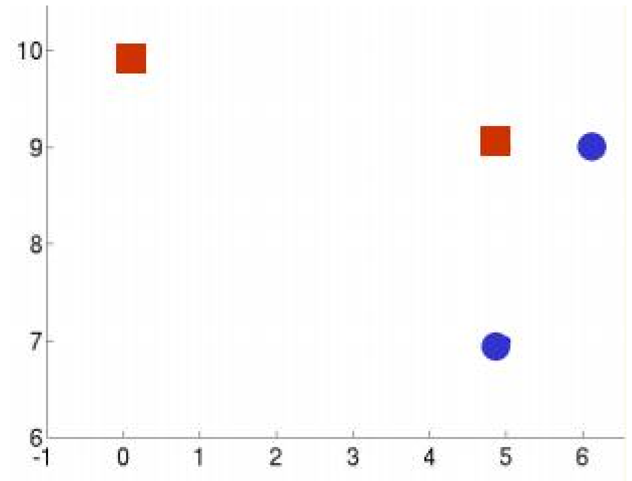
- Either by $\mathbf{e}^{(k)} = 0$ or one of its components is positive

Ho-Kashyap Procedure

- In the linearly separable case,
  - $\mathbf{e}^{(k)} = 0$, found solution, stop
  - one of components of $\mathbf{e}^{(k)}$ is positive, algorithm continues
- In non separable case,
  - $\mathbf{e}^{(k)}$ will have only negative components eventually, thus found proof of nonseparability
  - No bound on how many iteration need for the proof of nonseparability

Example

- Class 1: (6,9), (5,7)
- Class 2: (5,9), (0, 10)
- Matrix $Y = \begin{bmatrix} 1 & 6 & 9 \\ 1 & 5 & 7 \\ -1 & -5 & -9 \\ -1 & 0 & -10 \end{bmatrix}$
- Start with $a^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $b^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
- Use fixed learning η = 0.9
- At the start $Ya^{(1)} = \begin{bmatrix} 16 \\ 13 \\ -15 \\ -11 \end{bmatrix}$



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 $\mathbf{b}^{(2)}$ using $\mathbf{a}^{(1)}$ and $\mathbf{b}^{(1)}$

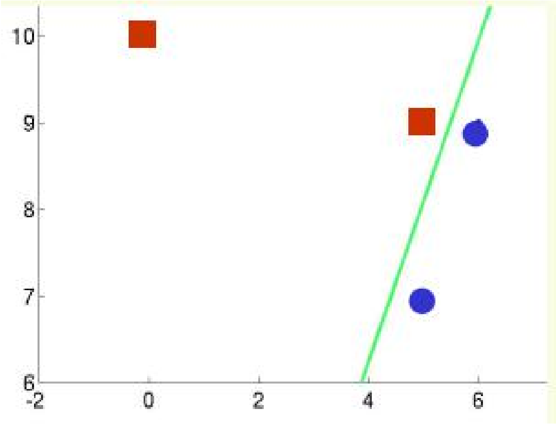
$$b^{(2)} = b^{(1)} + 0.9\left[e^{(1)} + |e^{(1)}|\right] = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + 0.9\left[\begin{bmatrix} 15 \\ 12 \\ -16 \\ -12 \end{bmatrix} + \begin{bmatrix} 15 \\ 12 \\ 16 \\ 12 \end{bmatrix}\right] = \begin{bmatrix} 28 \\ 22.6 \\ 1 \\ 1 \end{bmatrix}$$

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

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

Example

- Continue iterations until **Ya** > 0
  - In practice, continue until minimum component of **Ya** is less than 0.01



- After 104 iterations converged to solution

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

- **a** does gives a separating hyperplane

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

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 even 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

Q & A