Minimum Squared Error

LDF: Minimum Squared-Error Procedures

- Idea: convert to easier and better understood problem

Perceptron

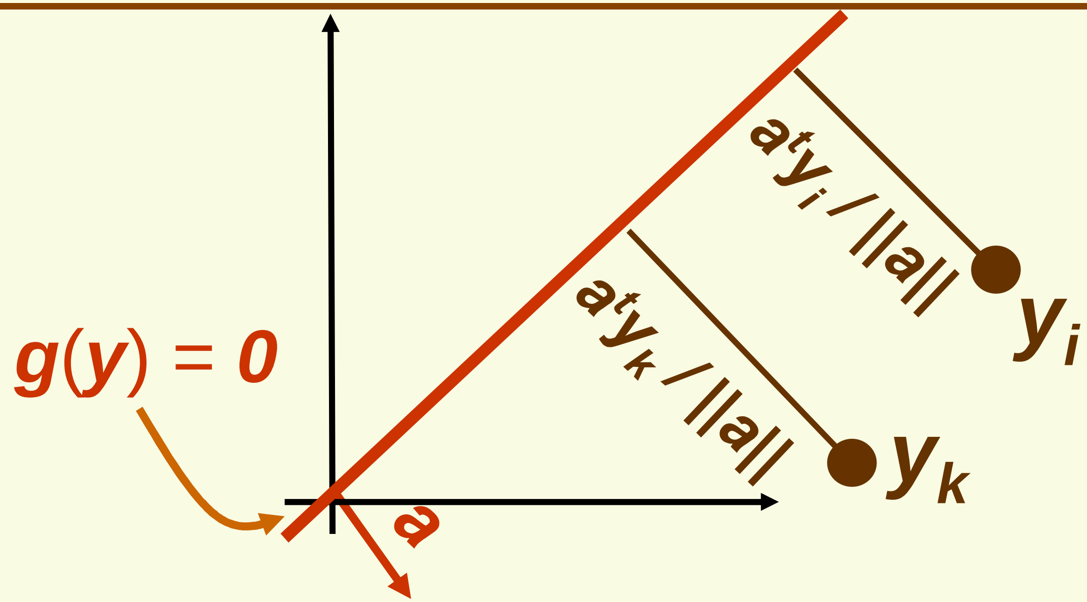
$\boldsymbol{a}^t\boldsymbol{y}_i > 0$ for all samples $\boldsymbol{y}_i$
solve system of linear inequalities

$\Downarrow$

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

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

LDF: MSE Margins



- Since we want $a^t y_i = b_i$, we expect sample y_i to be at distance b_i from the separating hyperplane (normalized by $||a||$)
- Thus $b_1, b_2, \dots, b_n$ 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 make b_i larger otherwise
- In the absence of any additional information, there are good reasons to set $b_1 = b_2 = \dots = b_n = 1$

LDF: MSE Matrix Notation

- Need to solve ***n*** equations
$$\begin{cases} \boldsymbol{a}^t \boldsymbol{y}_1 = \boldsymbol{b}_1 \\ \quad\vdots \\ \boldsymbol{a}^t \boldsymbol{y}_n = \boldsymbol{b}_n \end{cases}$$

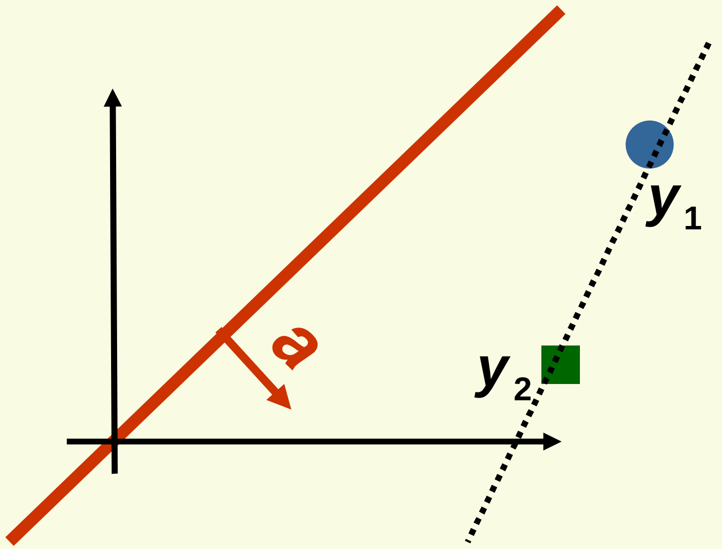
- Introduce matrix notation:

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

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

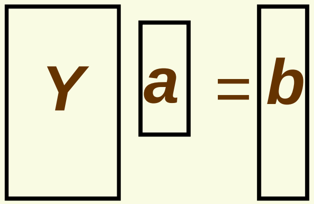
LDF: Exact Solution is Rare

- Thus need to solve a linear system $\boldsymbol{Ya} = \boldsymbol{b}$
  - $\boldsymbol{Y}$ is an $\boldsymbol{n}$ by ($\boldsymbol{d}$ +$\boldsymbol{1}$) matrix
- Exact solution can be found only if $\boldsymbol{Y}$ 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

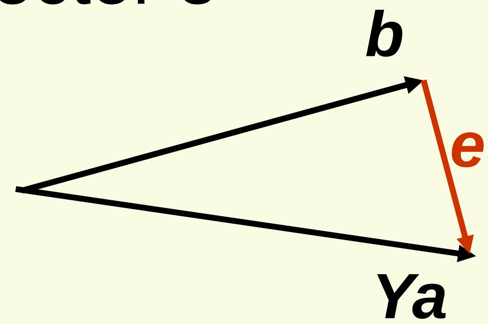


- Need ***Ya*** = ***b***, but no exact solution exists for an overdetermined system of equation
  - More equations than unknowns
- Find an approximate solution ***a***, that is $\boldsymbol{Ya \approx b}$
  - Note that approximate solution ***a*** ***does not*** necessarily give the separating hyperplane 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:

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

- 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:

$$\nabla J_s(a) = \begin{bmatrix} \frac{\partial J_s}{\partial a_0} \\ \vdots \\ \frac{\partial J_s}{\partial a_d} \end{bmatrix} = 2Y^t(Ya - b)$$

- Setting the gradient to 0:

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

LDF: Pseudo Inverse Solution

- 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:

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

pseudo inverse of ***Y***

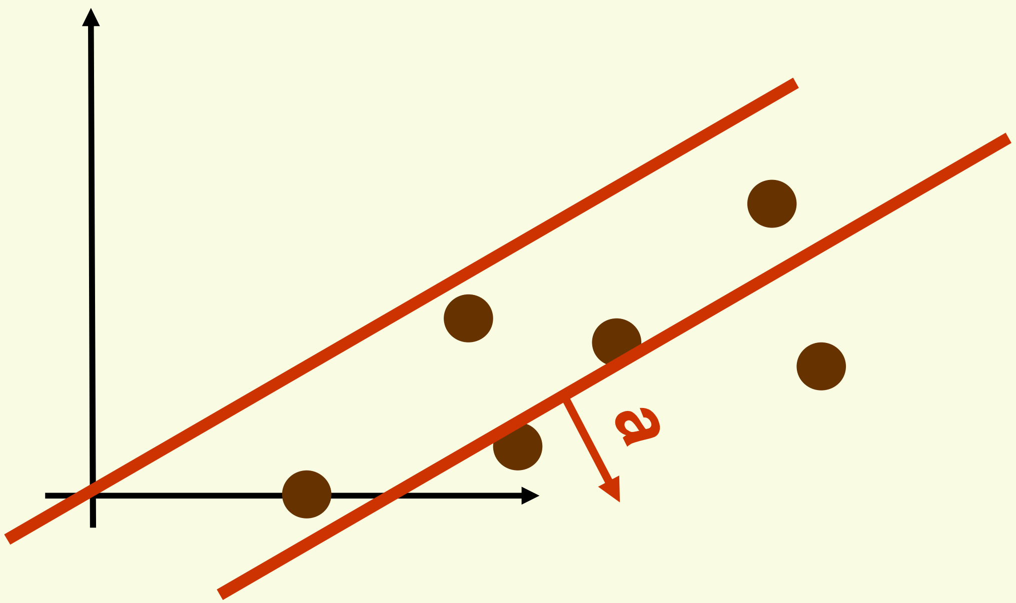
$$\left(\left(Y^tY\right)^{-1}Y^t\right)Y = \left(Y^tY\right)^{-1}\left(Y^tY\right) = I$$

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 $\boldsymbol{y_1},\ldots,\boldsymbol{y_n}$

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

$$\mathbf{1}_n = \left.\begin{bmatrix} \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix}\right\} \boldsymbol{n}$$



- 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 $\varepsilon_1, \ldots, \varepsilon_n$ are small relative to $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, 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 $\boldsymbol{a}$ 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 $\boldsymbol{Ya} \approx \boldsymbol{b} > \boldsymbol{0}$
- 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****

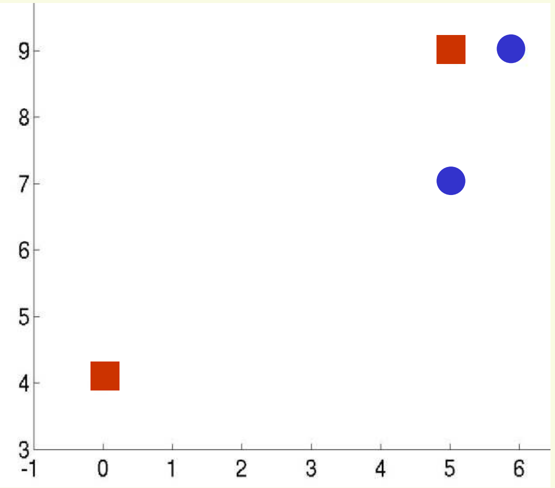
$$\arg\min_{\boldsymbol{a}} \left\| \boldsymbol{Ya} - \beta\boldsymbol{b} \right\|^2 = \arg\min_{\boldsymbol{a}} \beta^2 \left\| \boldsymbol{Y}(\boldsymbol{a}/\beta) - \boldsymbol{b} \right\|^2$$

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

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

LDF: Example

- Class 1: (6 9), (5 7)
- Class 2: (5 9), (0 4)



- Set vectors y_1, y_2 , y_3 , y_4 by adding extra feature and "normalizing"

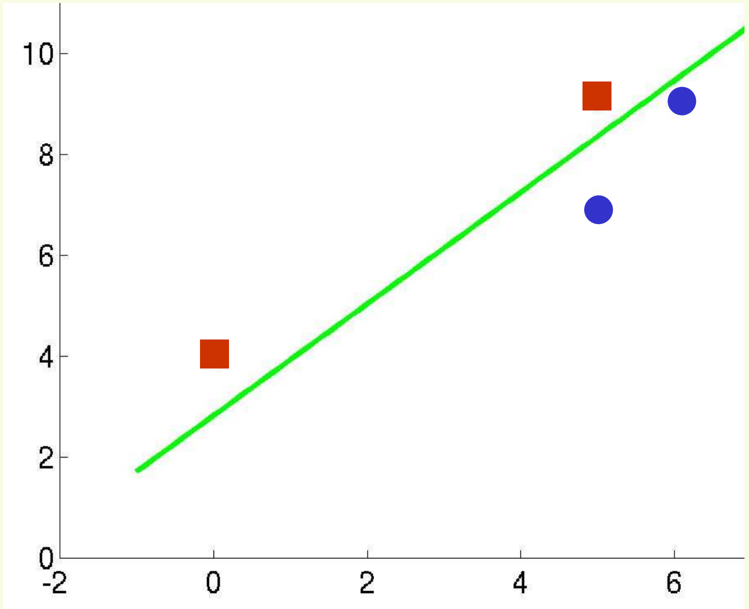
$$y_1 = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} \quad y_3 = \begin{bmatrix} -1 \\ -5 \\ -9 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ 0 \\ -4 \end{bmatrix}$$

- Matrix Y is then

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

LDF: Example

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



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

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

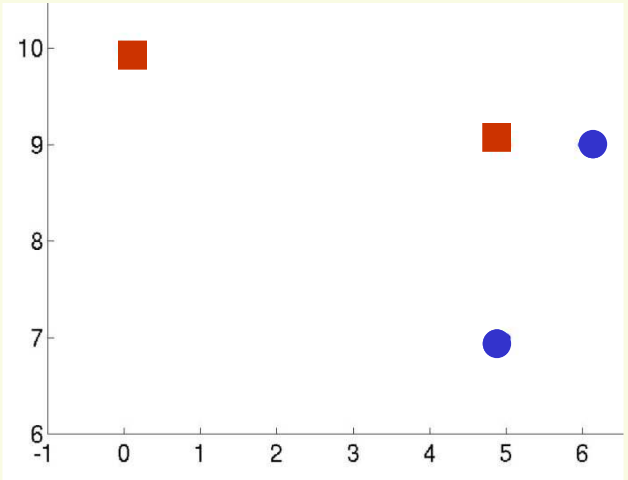
- Note ***a*** is an approximation to ***Ya*** = ***b***, since no exact solution exists

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

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

LDF: Example

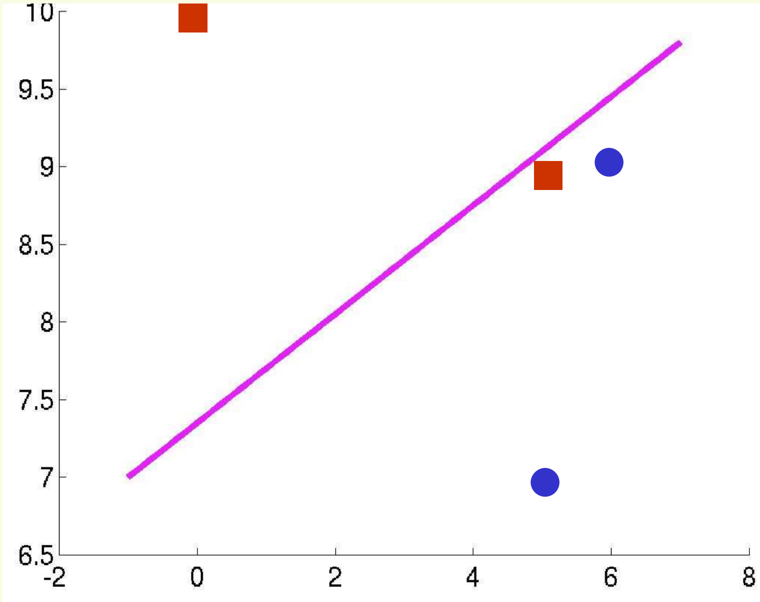
- Class 1: (6 9), (5 7)
- Class 2: (5 9), (0 10)
- The last sample is very far compared to others from the separating hyperplane



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

- 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

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

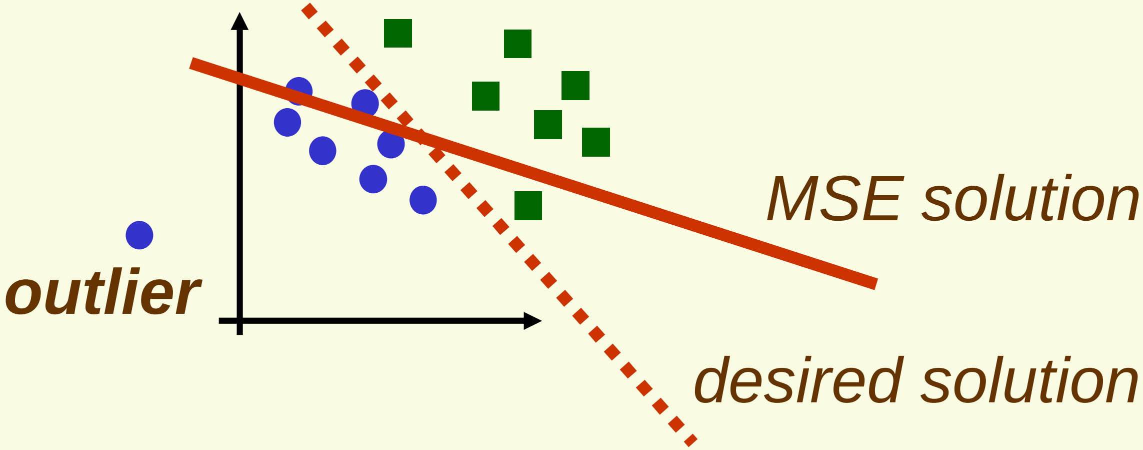
- Note ***a*** is an approximation to ***Ya*** = ***b***, since no exact solution exists

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

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

LDF: Example

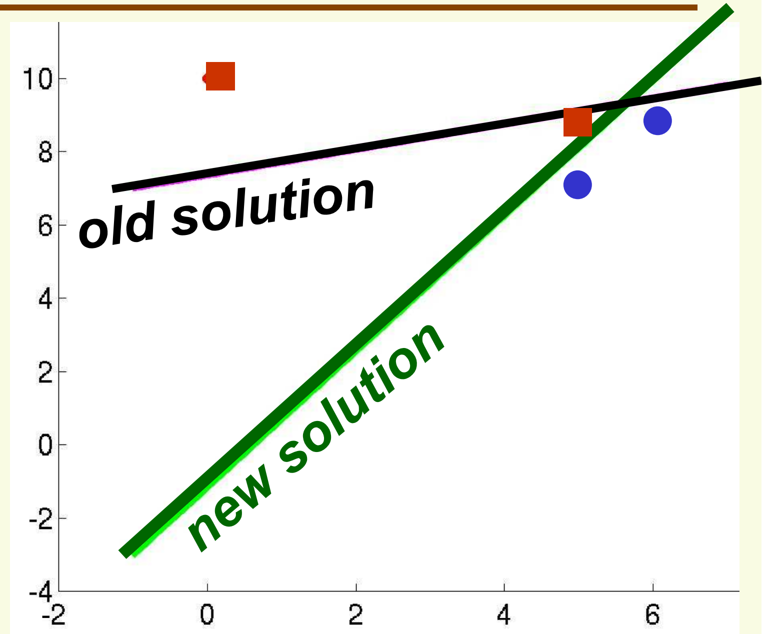
- MSE pays to much attention to isolated “noisy” examples (such examples are called outliers)



- 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 $\boldsymbol{Ya} > \boldsymbol{0}$

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

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

- Thus the update rule for gradient descent:

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

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

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

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

LDF: 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 $\boldsymbol{a}^s$ and positive $\boldsymbol{b}^s$ s.t.

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

- 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 ***b***:

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

LDF: Ho-Kashyap Procedure

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

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

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

- 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

$$a = (Y^{t}Y)^{-1}Y^{t}b$$

LDF: Ho-Kashyap Procedure

- Step 2: fix $\boldsymbol{a}$ and minimize $J_{HK}(\boldsymbol{a},\boldsymbol{b})$ with respect to $\boldsymbol{b}$
- We can't use $\boldsymbol{b} = \boldsymbol{Ya}$ because $\boldsymbol{b}$ has to be positive
- Solution: use modified gradient descent
- start with positive $\boldsymbol{b}$, follow negative gradient but refuse to decrease any components of $\boldsymbol{b}$
- This can be achieved by setting all the positive components of $\nabla_b J$ to $\boldsymbol{0}$
- Not doing steepest descent anymore, but we are still doing descent ***and*** ensure that $\boldsymbol{b}$ is positive

LDF: Ho-Kashyap Procedure

- The 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

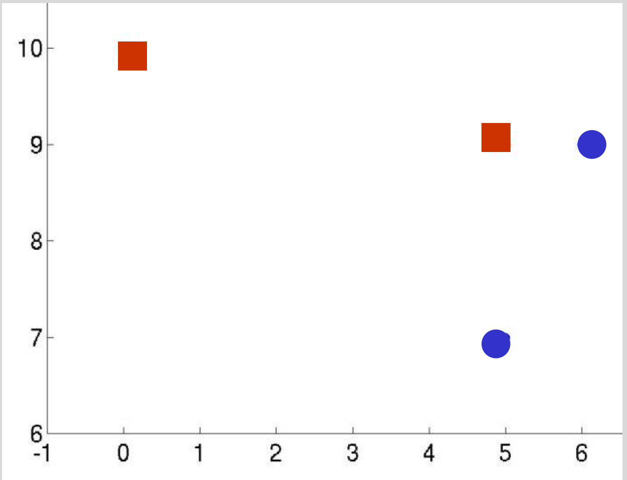
until $\boldsymbol{e}^{(k)} >= 0$ or $\boldsymbol{k} > \boldsymbol{k}_{max}$ or $\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)}$

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

LDF: Ho-Kashyap Procedure

- 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

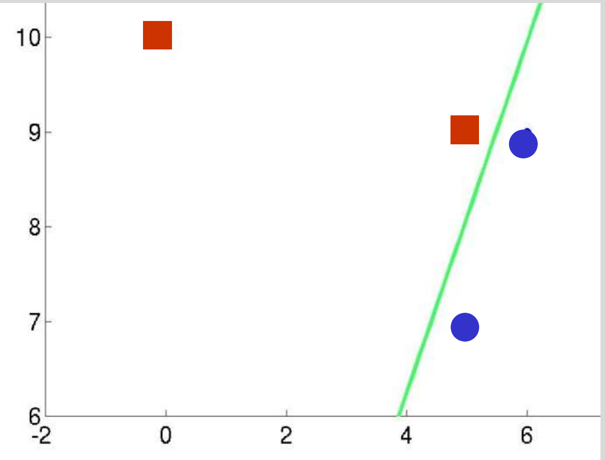
- Iteration 1:
  - $$e^{(1)} = Ya^{(1)} - b^{(1)} = \begin{bmatrix} 16 \\ 13 \\ -15 \\ -11 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 15 \\ 12 \\ -16 \\ -12 \end{bmatrix}$$
  - solve for $b^{(2)}$ using $a^{(1)}$ and $b^{(1)}$

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

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

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

LDF: Ho-Kashyap Procedure Example



- Continue iterations until $Ya > 0$
  - In practice, continue until minimum component of Ya is less then 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: MSE for Multiple Classes

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

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

- Given ***x***, assign class c_i if

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

- 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: MSE for Multiple Classes

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

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

- 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 $\boldsymbol{1}$:

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

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 $\boldsymbol{A}$

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

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

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

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 $\boldsymbol{Y^tY}$ 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