## CS442/542b: Artificial Intelligence II
## Prof. Olga Veksler

Lecture 4: Machine Learning

**Linear Classifier**

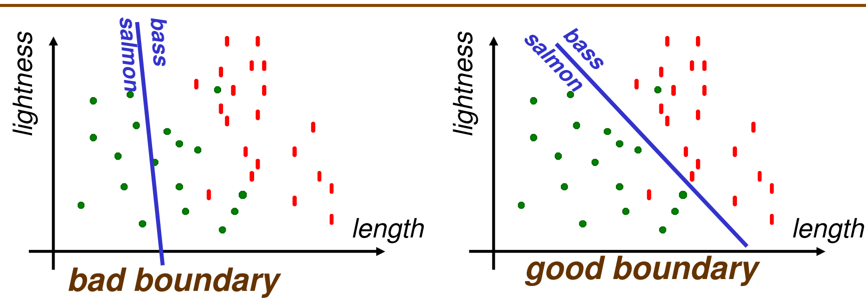
## Outline

- Linear Classification or Linear Discriminant Functions
  - Introduction
  - 2 classes
  - Multiple classes
  - Optimization with gradient descent
  - Perceptron Criterion Function
    - Batch perceptron rule
    - Single sample perceptron rule
  - Minimum Squared Error (MSE) rule
    - Pseudoinverse
    - Gradient descent (Widrow-Hoff Procedure)

## Discriminant Function

- Consider 2 class case, and let's denote them
  - class 1
  - class -1
- Want to build f(X,W) to give correct class of sample X
- Let f(X,W) = sign(g(X,W))
  - 1 if g(X,W) is positive
  - -1 if g(X,W) is negative
- g(X,W) is called the ***discriminant function***

## Linear Discriminant Functions: Basic Idea




- Have samples from 2 classes $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}$
- Assume 2 classes can be classified by a linear discriminant function g(X,W) with some unknown parameters W
- Fit the "best" g(X,W) to data by optimizing over parameters W
- What is best g(X,W)?
  - simplest idea: minimize classification error on training data

## LDF: Introduction

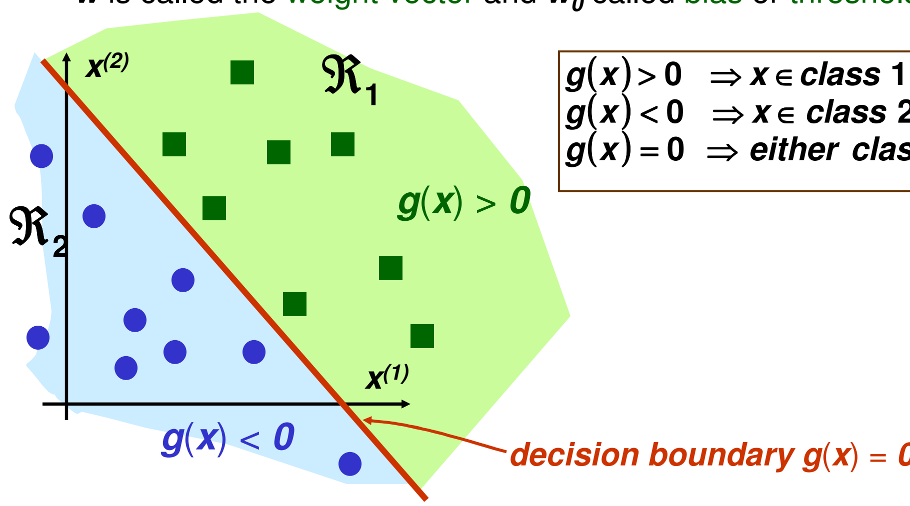
- Discriminant functions can be more general than linear
- For now, we will study linear discriminant functions
  - Simple model (should try simpler models first)
  - Analytically tractable
- Linear Discriminant functions are optimal for certain type of data
  - Gaussian distributions with equal covariance (don't worry if you don't know what a Gaussian is)
- May not be optimal for other data distributions, but they are very simple to use

## LDF: 2 Classes

- A discriminant function is linear if it can be written as

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0$$

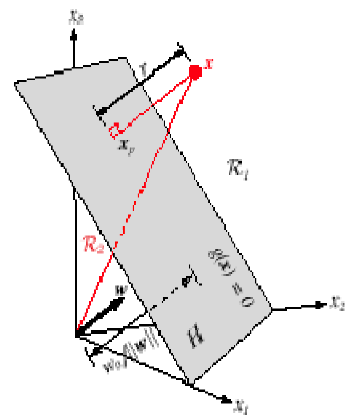
  - $\mathbf{w}$ is called the weight vector and $w_0$ called bias or threshold




$$\begin{aligned} g(\mathbf{x}) > 0 &\Rightarrow \mathbf{x} \in \textit{class } 1 \\ g(\mathbf{x}) < 0 &\Rightarrow \mathbf{x} \in \textit{class } 2 \\ g(\mathbf{x}) = 0 &\Rightarrow \textit{either class} \end{aligned}$$

## LDF: 2 Classes

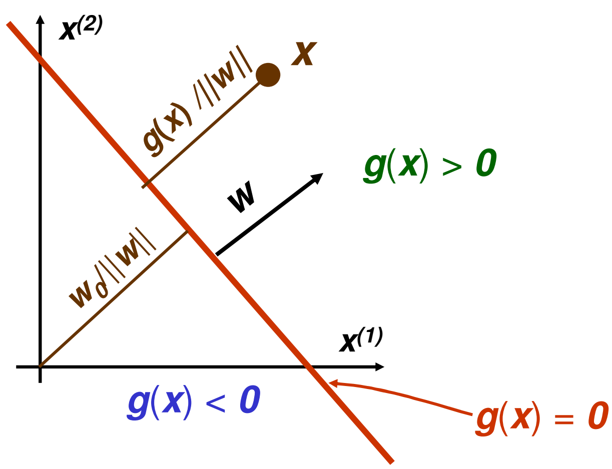
- Decision boundary $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{w}^t\boldsymbol{x} + \boldsymbol{w}_0=0$ is a ***hyperplane***
  - set of vectors x which for some scalars $\alpha_0,\ldots,\alpha_d$ satisfy $\alpha_0 + \alpha_1 x^{(1)} + \ldots + \alpha_d x^{(d)} = 0$
- A hyperplane is
  - a point in 1D
  - a line in 2D
  - a plane in 3D




## LDF: 2 Classes

$$g(x) = w^t x + w_0$$

- $\boldsymbol{w}$ determines orientation of the decision hyperplane
- $\boldsymbol{w_0}$ determines location of the decision surface



## LDF: 2 Classes

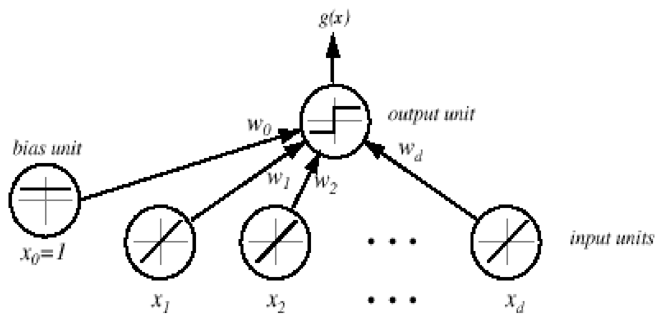



FIGURE 5.1. A simple linear classifier having $d$ input units, each corresponding to the values of the components of an input vector. Each input feature value $x_i$ is multiplied by its corresponding weight $w_i$; the effective input at the output unit is the sum all these products, $\sum w_i x_i$. We show in each unit its effective input-output function. Thus each of the $d$ input units is linear, emitting exactly the value of its corresponding feature value. The single bias unit unit always emits the constant value 1.0. The single output unit emits a +1 if $\mathbf{w}^t\mathbf{x} + w_0 > 0$ or a −1 otherwise. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

## LDF: Many Classes

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

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

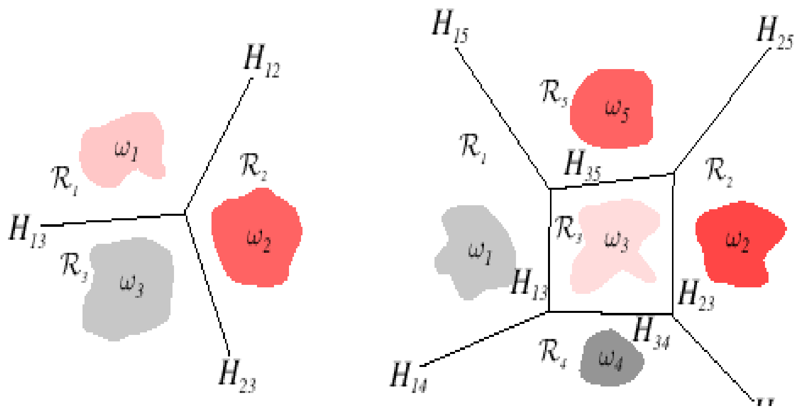
- Given ***x***, assign class $c_i$ if

$$g_i(x) \ge g_j(x) \qquad \forall j \ne i$$

- 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 $R_i$

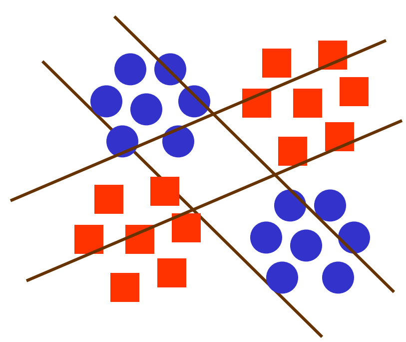
## LDF: Many Classes



- Decision regions for a linear machine are ***convex***
  - In particular, decision regions must be spatially contiguous

## LDF: Many Classes

- Thus applicability of linear machine to mostly limited to unimodal distributions
  - Most data is concentrated around one point in space
- Example:



  - need non-contiguous decision regions
  - thus linear machine will fail

## LDF: Augmented feature vector

- Linear discriminant function: $g(x) = w^t x + w_0$
- Can rewrite it: $g(x) = \underbrace{\begin{bmatrix} w_0 & w^t \end{bmatrix}}_{\text{new weight vector } a} \underbrace{\begin{bmatrix} 1 \\ x \end{bmatrix}}_{\text{new feature vector } y} = a^t y = g(y)$
- $y$ is called the ***augmented feature vector***
- Added a dummy dimension to get a completely equivalent new ***homogeneous*** problem

*old problem*

$$g(x) = w^t x + w_0$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

*new problem*

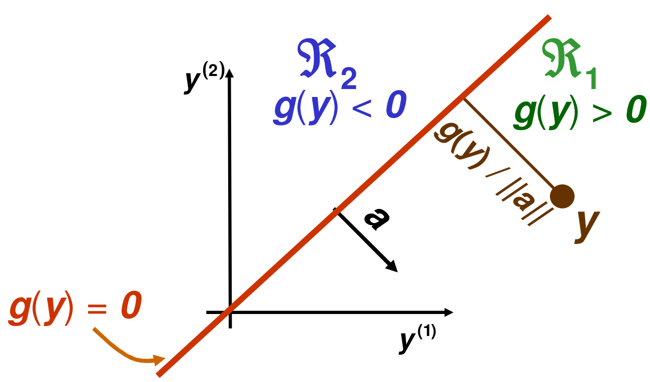
$$g(y) = a^t y$$

$$\begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$

## LDF: Augmented feature vector

- Feature augmenting is done for simpler notation
- From now on we always assume that we have augmented feature vectors
  - Given samples $x_1, \ldots, x_n$ convert them to augmented samples $y_1, \ldots, y_n$ by adding a new dimension of value 1

$$y_i = \begin{bmatrix} 1 \\ x_i \end{bmatrix}$$



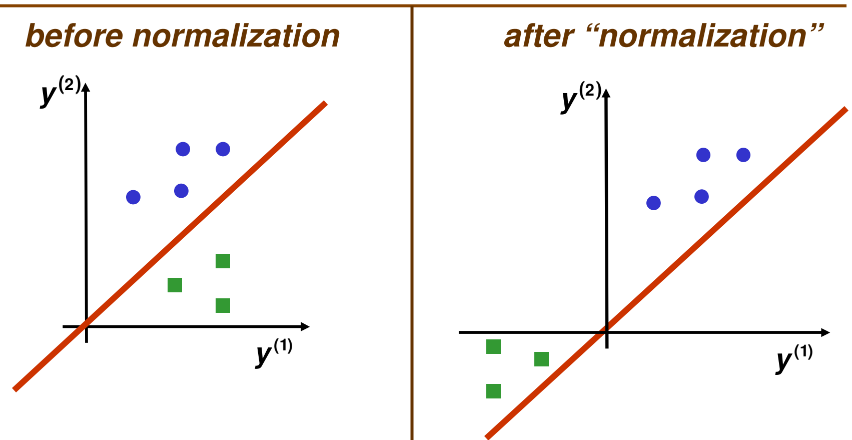
## LDF: Training Error

- For the rest of the lecture, assume we have 2 classes
- Samples $y_1, \ldots, y_n$ some in class 1, some in class 2
- Use these samples to determine weights $a$ in the discriminant function $g(y) = a^t y$
- What should be our criterion for determining $a$?
  - For now, suppose we want to minimize the training error (that is the number of misclassifed samples $y_1, \ldots, y_n$ )
- Recall that $$g(y_i) > 0 \Rightarrow y_i \text{ classified } c_1$$ $$g(y_i) < 0 \Rightarrow y_i \text{ classified } c_2$$
- Thus training error is *0* if $$\begin{cases} g(y_i) > 0 & \forall y_i \in c_1 \\ g(y_i) < 0 & \forall y_i \in c_2 \end{cases}$$

## LDF: Problem "Normalization"

- Thus training error is *0* if $$\begin{cases} a^t y_i > 0 & \forall y_i \in c_1 \\ a^t y_i < 0 & \forall y_i \in c_2 \end{cases}$$
- Equivalently, training error is *0* if $$\begin{cases} a^t y_i > 0 & \forall y_i \in c_1 \\ a^t(-y_i) > 0 & \forall y_i \in c_2 \end{cases}$$
- This suggest problem "normalization":
  1. Replace all examples from class $c_2$ by their negative $$y_i \to -y_i \qquad \forall y_i \in c_2$$
  2. Seek weight vector $a$ s.t. $$a^t y_i > 0 \qquad \forall y_i$$
  - If such $a$ exists, it is called a *separating* or *solution* vector
  - Original samples $x_1, \ldots, x_n$ can indeed be separated by a line then

## LDF: Problem "Normalization"

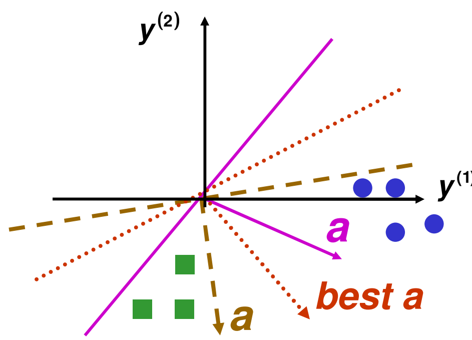



Seek a hyperplane that separates patterns from different categories ⇦ Seek hyperplane that puts *normalized* patterns on the same (positive) side

## LDF: Solution Region

- Find weight vector ***a*** s.t. for all samples $y_1,\ldots, y_n$

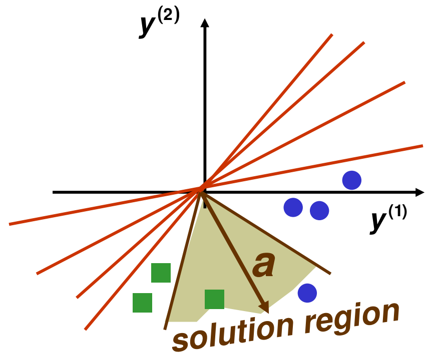
$$a^t y_i = \sum_{k=0}^{d} a_k y_i^{(k)} > 0$$




- In general, there are many such solutions ***a***

## LDF: Solution Region

- ***Solution region*** for ***a***: set of all possible solutions
  - defined in terms of normal ***a*** to the separating hyperplane




## Optimization

- Need to minimize a function of many variables

$$J(x) = J(x_1, \ldots, x_d)$$

- We know how to minimize ***J***(***x***)
  - Take partial derivatives and set them to zero

$$\begin{bmatrix} \frac{\partial}{\partial x_1} J(x) \\ \vdots \\ \frac{\partial}{\partial x_d} J(x) \end{bmatrix} = \nabla J(x) = 0$$

gradient

- However solving analytically is not always easy
  - Would you like to solve this system of nonlinear equations?

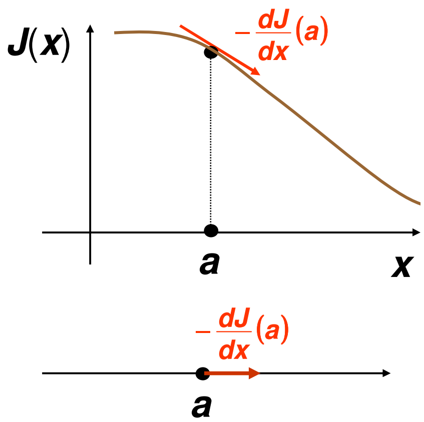
$$\begin{cases} \sin(x_1^2 + x_2^3) + e^{x_4^2} = 0 \\ \cos(x_1^2 + x_2^3) + \log(x_5^3)^{x_4^2} = 0 \end{cases}$$

  - Sometimes it is not even possible to write down an analytical expression for the derivative, we will see an example later today

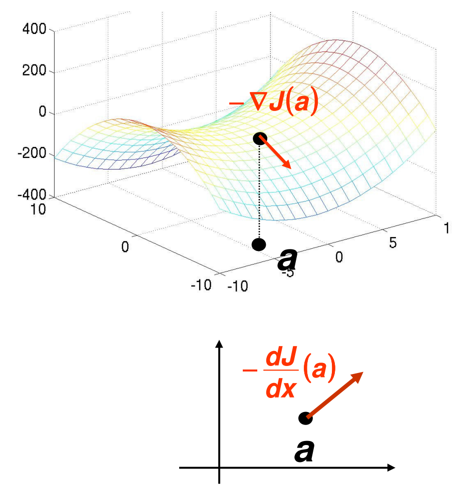
## Optimization: Gradient Descent

- Gradient $\nabla \boldsymbol{J}(\boldsymbol{x})$ points in direction of steepest increase of $\boldsymbol{J}(\boldsymbol{x})$, and $-\nabla \boldsymbol{J}(\boldsymbol{x})$ in direction of steepest decrease

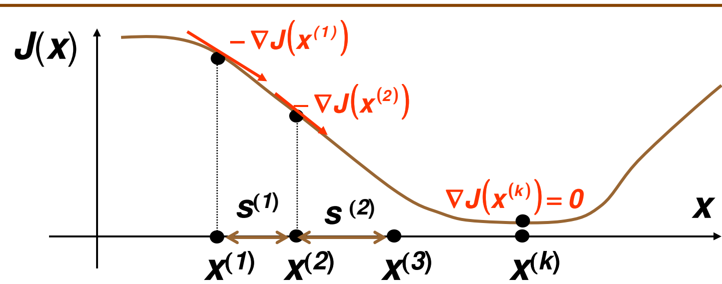
***one dimension***




***two dimensions***




## Optimization: Gradient Descent




***Gradient Descent*** for minimizing any function $\boldsymbol{J}(\boldsymbol{x})$

set $\boldsymbol{k} = \boldsymbol{1}$ ***and*** $\boldsymbol{x}^{(1)}$ to some initial guess for the weight vector

**while** $\eta^{(k)} \left| \nabla \boldsymbol{J}(\boldsymbol{x}^{(k)}) \right| > \varepsilon$

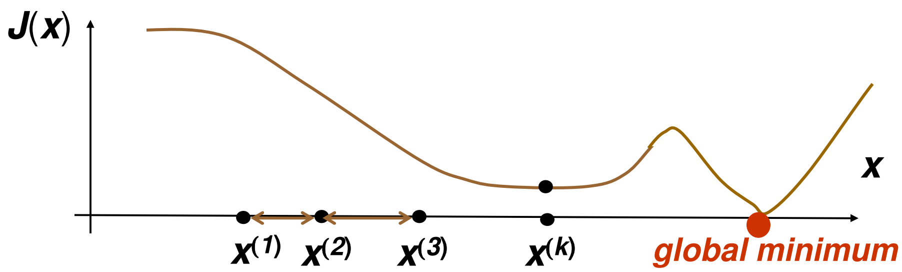
***choose learning rate*** $\eta^{(k)}$

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \eta^{(k)} \nabla \boldsymbol{J}(\boldsymbol{x})$ ***(update rule)***

$\boldsymbol{k} = \boldsymbol{k} + \boldsymbol{1}$

## Optimization: Gradient Descent

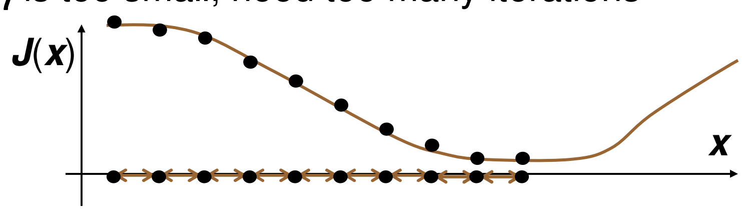
- Gradient descent is guaranteed to find only a local minimum



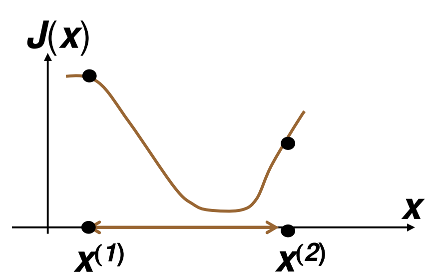

- Nevertheless gradient descent is very popular because it is simple and applicable to any function

## Optimization: Gradient Descent

- Main issue: how to set parameter $\eta$ (***learning rate*** )
- If $\eta$ is too small, need too many iterations




- If $\eta$ is too large may overshoot the minimum and possibly never find it (if we keep overshooting)



## *LDF: Criterion Function*

- Find weight vector $\boldsymbol{a}$ s.t. for all samples $\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}$

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

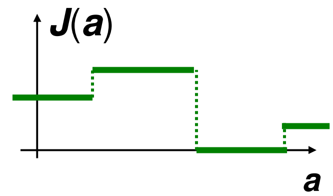
- Need criterion function $\boldsymbol{J}(\boldsymbol{a})$ which is minimized when $\boldsymbol{a}$ is a solution vector
- Let $\boldsymbol{Y_M}$ be the set of examples misclassified by $\boldsymbol{a}$

$$Y_M(a) = \{ \text{sample } y_i \text{ s.t. } a^t y_i < 0 \}$$

- First natural choice: number of misclassified examples

$$J(a) = |Y_M(a)|$$

  - piecewise constant, gradient descent is useless

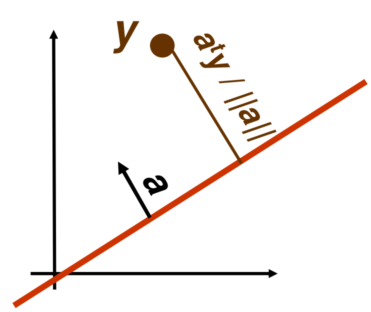



## *LDF: Perceptron Criterion Function*

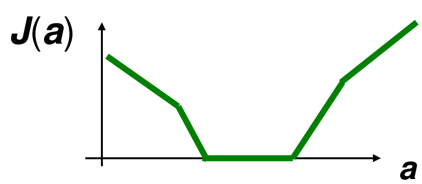
- Better choice: ***Perceptron*** criterion function

$$J_p(a) = \sum_{y \in Y_M} (-a^t y)$$

- If $\boldsymbol{y}$ is misclassified, $a^t y \leq 0$
- Thus $J_p(a) \geq 0$
- $\boldsymbol{J_p}(\boldsymbol{a})$ is -||$\boldsymbol{a}$|| times sum of distances of misclassified examples to decision boundary




- $\boldsymbol{J_p}(\boldsymbol{a})$ is piecewise linear and thus suitable for gradient descent



## LDF: Perceptron Batch Rule

$$J_p(a) = \sum_{y \in Y_M} (-a^t y)$$

- Gradient of $J_p(a)$ is $\nabla J_p(a) = \sum_{y \in Y_M} (-y)$
  - $Y_M$ are samples misclassified by $a^{(k)}$
  - It is not possible to solve $\nabla J_p(a) = 0$ analytically because of $Y_M$
- Update rule for gradient descent: $x^{(k+1)} = x^{(k)} - \eta^{(k)} \nabla J(x)$
- Thus *gradient decent batch update rule* for $J_p(a)$ is:

$$a^{(k+1)} = a^{(k)} + \eta^{(k)} \sum_{y \in Y_M} y$$

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

## LDF: Perceptron Single Sample Rule

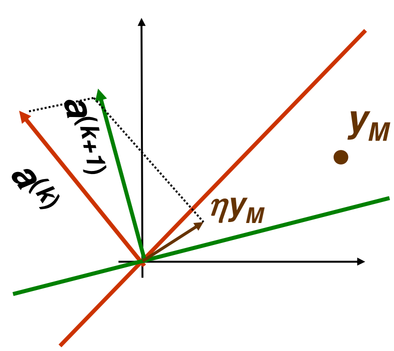
- Thus *gradient decent single sample rule* for $J_p(a)$ is:

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

  - note that $y_M$ is one sample misclassified by $a^{(k)}$
  - must have a consistent way of visiting samples
- Geometric Interpretation:
  - $y_M$ misclassified by $a^{(k)}$

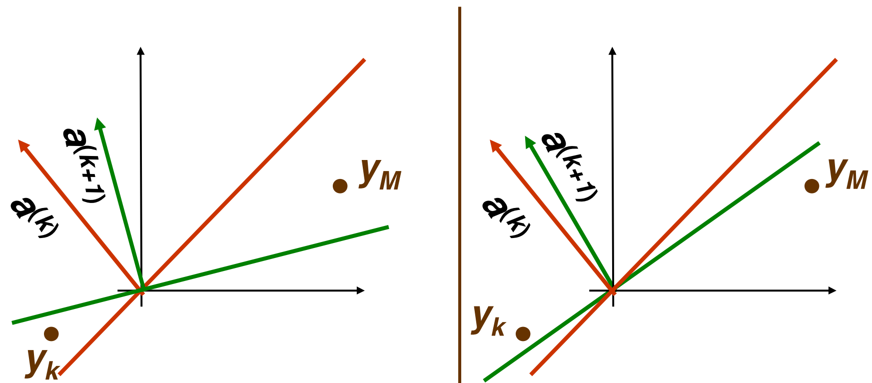
$$\left(a^{(k)}\right)^t y_M \le 0$$

  - $y_M$ is on the wrong side of decision hyperplane
  - adding $\eta y_M$ to $a$ moves new decision hyperplane in the right direction with respect to $y_M$



## LDF: Perceptron Single Sample Rule

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




$\eta$ is too large, previously correctly classified sample $y_k$ is now misclassified

$\eta$ is too small, $y_M$ is still misclassified

## LDF: 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 grade *A*
- ***class 2***: students who get grade *F*

## LDF Example: Augment feature vector

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

- convert samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ to augmented samples $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ by adding a new dimension of value 1

## LDF: Perform "Normalization"

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

- Replace all examples from class $\boldsymbol{c}_2$ by their negative

$$\boldsymbol{y}_i \rightarrow -\boldsymbol{y}_i \qquad \forall \boldsymbol{y}_i \in \boldsymbol{c}_2$$

- Seek weight vector $\boldsymbol{a}$ s.t. $\boldsymbol{a}^t \boldsymbol{y}_i > \mathbf{0} \qquad \forall \boldsymbol{y}_i$

## LDF: Use Single Sample Rule

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

- Sample is misclassified if $\mathbf{a}^t\mathbf{y}_i = \sum_{k=0}^{4} \mathbf{a}_k\mathbf{y}_i^{(k)} < 0$
- gradient descent single sample rule: $\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \eta^{(k)}\mathbf{y}_M$
- Set ***fixed*** learning rate to $\eta^{(k)}$= 1: $\boxed{\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \mathbf{y}_M}$

## LDF: Gradient decent Example

- set equal initial weights $\mathbf{a}^{(1)}$=[0.25, 0.25, 0.25, 0.25]
- visit all samples sequentially, modifying the weights for after finding a misclassified example

| ***name*** | $\mathbf{a}^t\mathbf{y}$ | ***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

$$\mathbf{a}^{(2)} = \mathbf{a}^{(1)} + \mathbf{y}_M = [\mathbf{0.25} \;\; \mathbf{0.25} \;\; \mathbf{0.25} \;\; \mathbf{0.25} \;\; \mathbf{0.25}] +$$
$$+[-\mathbf{1} \;\; -\mathbf{1} \;\; -\mathbf{1} \;\; -\mathbf{1} \;\; -\mathbf{1}] =$$
$$=[-\mathbf{0.75} \;\; -\mathbf{0.75} \;\; -\mathbf{0.75} \;\; -\mathbf{0.75} \;\; -\mathbf{0.75}]$$

## LDF: Gradient decent 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

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

## LDF: Gradient decent Example

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

| 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

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

## LDF: Gradient decent 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***:

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

## LDF: Gradient decent Example

- Converting back to the original features ***x***:

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

good attendance — tall — sleeps in class — chews gum

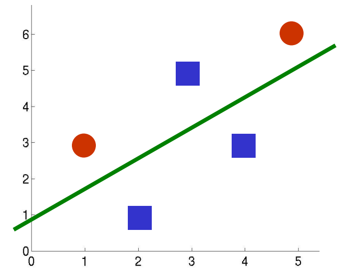
- This is just one possible solution vector
- If we started with weights $a^{(1)}$=[0,0.5, 0.5, 0, 0], solution would be [-1,1.5, -0.5, -1, -1]

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

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

## LDF: Nonseparable Example

- Suppose we have 2 features and 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

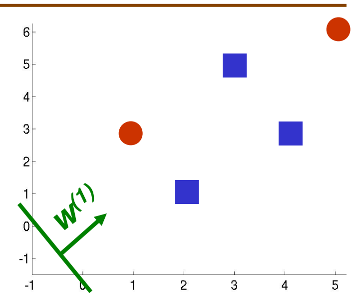


- Still would like to get approximate separation by a line, good choice is shown in green
  - some samples may be “noisy”, and it’s ok if they are on the wrong side of the line
- Get $\boldsymbol{y_1}$, $\boldsymbol{y_2}$, $\boldsymbol{y_3}$, $\boldsymbol{y_4}$ by adding extra feature and “normalizing”

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

## LDF: Nonseparable Example

- Let’s apply Perceptron single sample algorithm
- initial equal weights $\boldsymbol{a}^{(1)} = [1\ 1\ 1]$
  - this is line $x^{(1)}+x^{(2)}+1=0$
- fixed learning rate $\eta = 1$




$$\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} + \boldsymbol{y}_M$$

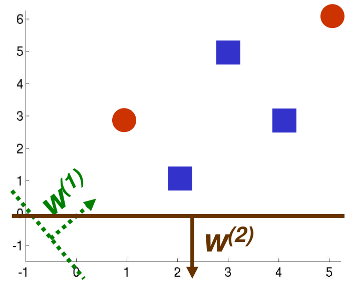
$$\boldsymbol{y_1} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \boldsymbol{y_2} = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad \boldsymbol{y_3} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad \boldsymbol{y_4} = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad \boldsymbol{y_5} = \begin{bmatrix}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

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

## LDF: Nonseparable Example

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

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




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

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

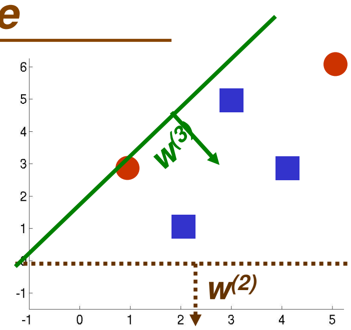
- $y^t_5\, a^{(2)}$=[0 0 -2]*[-1 -5 -6]$^t$ = 12 > 0 ✔
- $y^t_1\, a^{(2)}$=[0 0 -2]*[1 2 1]$^t$ < 0

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

## LDF: Nonseparable Example

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

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



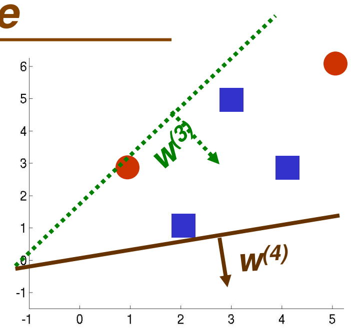

- $y^t_2\, a^{(3)}$=[1 4 3]*[1 2 -1]$^t$ =6 > 0 ✔
- $y^t_3\, a^{(3)}$=[1 3 5]*[1 2 -1]$^t$ > 0 ✔
- $y^t_4\, a^{(3)}$=[-1 -1 -3]*[1 2 -1]$^t$ = 0

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

## LDF: Nonseparable Example

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

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




- $y^t_2 a^{(3)}$=[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

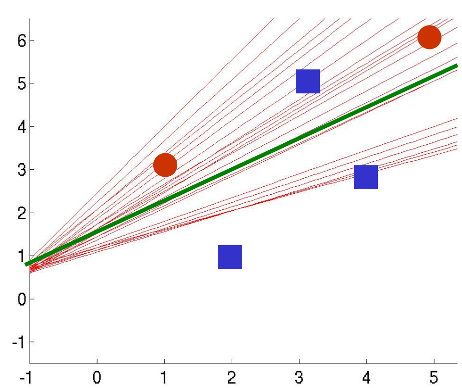
$$a^{(4)} = a^{(3)} + y_M = [1 \ \ 2 \ -1] + [-1 \ -1 \ -3] = [0 \ \ 1 \ -4]$$

## LDF: Nonseparable Example

- we can continue this forever
  - there is no solution vector ***a*** satisfying for all ***i***

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

- need to stop but at a good point:



- solutions at iterations 900 through 915. Some are good some are not.
- How do we stop at a good solution?

## LDF: Convergence of Perceptron rules

- If classes are linearly separable, and use fixed learning rate, that is for some constant $c$, $\eta^{(k)} = c$
  - *both 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:
  - algorithm does not stop, it keeps looking for solution which does not exist
  - by choosing appropriate learning rate, can always ensure convergence: $\eta^{(k)} \rightarrow 0 \;\; as \;\; k \rightarrow \infty$
  - for example inverse linear learning rate: $\eta^{(k)} = \dfrac{\eta^{(1)}}{k}$
  - 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

## LDF: 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<br>▪ Concentrates more than necessary on any isolated “noisy” training examples |

## LDF: Minimum Squared-Error Procedures

- Idea: convert to easier and better understood problem

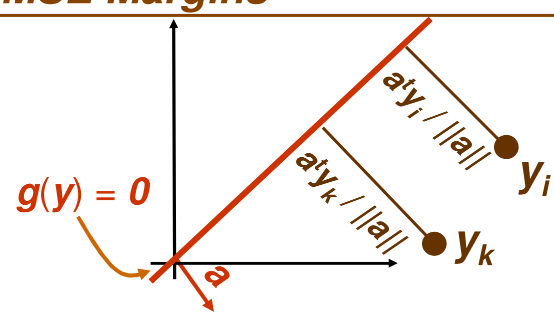
$a^t y_i > 0$ for all samples $y_i$
solve system of linear inequalities

⇓

$a^t y_i = b_i$ for all samples $y_i$
solve system of linear equations

- MSE procedure
  - Choose **positive** constants $b_1, b_2, \dots, b_n$
  - try to find weight vector $a$ s.t. $a^t y_i = b_i$ for all samples $y_i$
  - If we can find weight vector $a$ such that $a^t y_i = b_i$ for all samples $y_i$, then $a$ is a solution because $b_i$'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

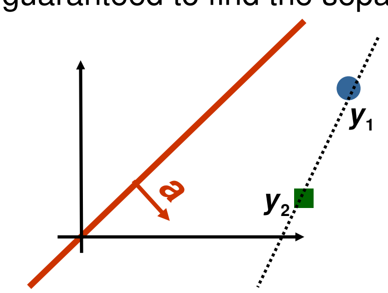
- Need to solve $n$ equations $\begin{cases} a^t y_1 = b_1 \\ \vdots \\ a^t y_n = b_n \end{cases}$
- Using matrix notation:

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

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

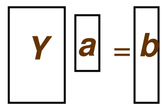
## LDF: Exact Solution is Rare

- 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 $Y^{-1}$ exists
  - $a = Y^{-1}b$
  - (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

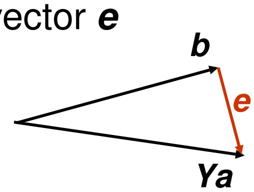



- Need ***Ya*** = ***b***, but no exact solution exists for an overdetermined system of equation
  - More equations than unknowns
- Find an approximate solution ***a***, that is $\boldsymbol{Ya} \approx \boldsymbol{b}$
  - Note that approximate solution ***a*** ***does not*** necessarily give the separating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

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

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

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

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

## LDF: Pseudo Inverse Solution

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

- Setting the gradient to 0:

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

- 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:

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

*pseudo inverse of* $Y$

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

- In matlab, simply **a=Y\b**

## 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 $\boldsymbol{Ya} = \begin{bmatrix} b_1 + \varepsilon_1 \\ \vdots \\ b_n + \varepsilon_n \end{bmatrix}$ where $\varepsilon$ 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, $\varepsilon_i$ may be large and negative, for some $i$, 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* give separating hyperplane
- But it will give a "reasonable" hyperplane

## LDF: Minimum Squared-Error Procedures

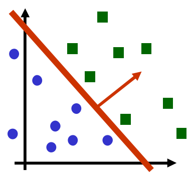
- We are free to choose $\boldsymbol{b}$. May be tempted to make $\boldsymbol{b}$ large as a way to insure $\boldsymbol{Ya} \approx \boldsymbol{b} > \boldsymbol{0}$
- Does not work
  - Let $\beta$ be a scalar, let's try $\beta \boldsymbol{b}$ instead of $\boldsymbol{b}$
  - if $\boldsymbol{a}^*$ is a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

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

  - thus if for some $i$th element of $\boldsymbol{Ya}$ 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 $\boldsymbol{b}$ matters, but not the size of each individual component

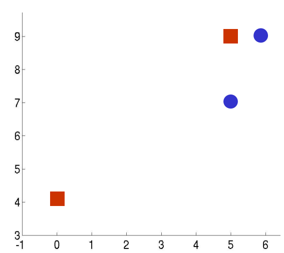
## LDF: How to choose b in MSE Procedure?

- So far we assumed that constants $b_1, b_2, \ldots, b_n$ are positive but otherwise arbitrary
- Good choice is $b_1 = b_2 = \ldots = b_n = 1$.
  - Interesting theoretical properties that we are not going to look at



## 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"



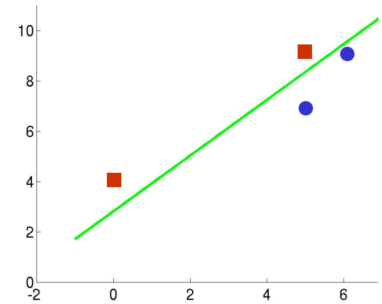

$$y_1 = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} \quad y_3 = \begin{bmatrix} -1 \\ -5 \\ -9 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ 0 \\ -4 \end{bmatrix}$$

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

## LDF: Example

- Choose $b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
- In matlab, $a = Y \backslash b$ solves the least squares problem

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



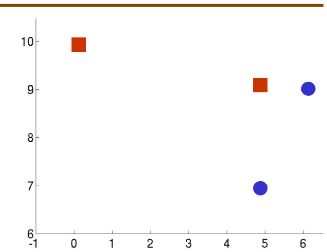

- Note $a$ is an approximation to $Ya = b$, since no exact solution exists

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

- 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

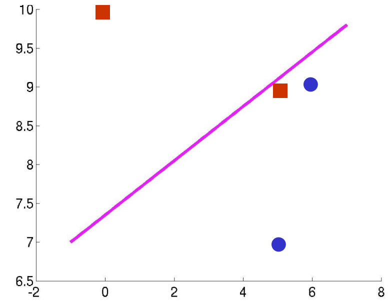



$$y_1 = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix} \quad y_2 = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} \quad y_3 = \begin{bmatrix} -1 \\ -5 \\ -9 \end{bmatrix} \quad y_4 = \begin{bmatrix} -1 \\ 0 \\ -10 \end{bmatrix}$$

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

## LDF: Example

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




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

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

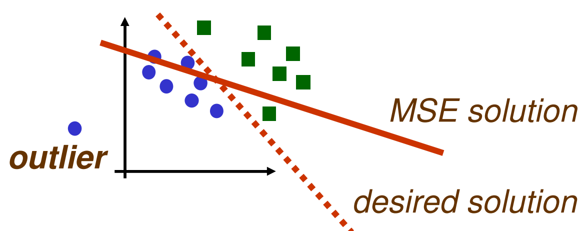
- Note $a$ is an approximation to $Ya = b$, since no exact solution exists

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

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

## LDF: Example

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

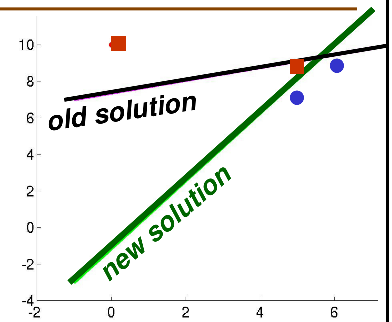



- No problems with convergence though, and solution it gives ranges from reasonable to good

## LDF: Example

- we know that 4th point is far far from separating hyperplane
  - In practice look at points which are furthest from the decision boundary

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




- Set $\mathbf{b_i}$ larger for such points: $\boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 10 \end{bmatrix}$
- In Matlab, solve $\boldsymbol{a} = \boldsymbol{Y} \backslash \boldsymbol{b}$

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

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

## LDF: Gradient Descent for MSE solution

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

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

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

## LDF: Widrow-Hoff Procedure

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

- Thus the update rule for gradient descent:

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

  - If $\eta^{(k)} = \eta^{(1)} / k$ weight vector $a^{(k)}$ converges to the MSE solution ***a***, that is $Y^t(Ya-b)=0$
- *Widrow-Hoff procedure* reduces storage requirements by considering single samples sequentially:

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

## LDF: MSE for Multiple Classes

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

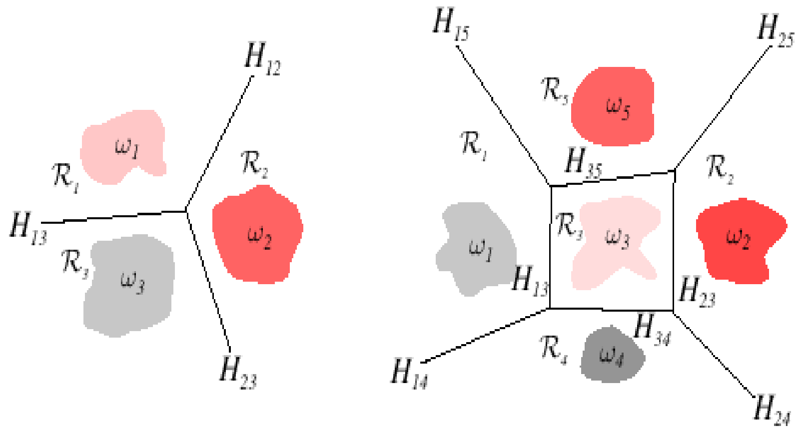
$$g_i(x) = w_i^t x + w_{i0} \qquad i = 1,\ldots,m$$

- 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: Many Classes




## LDF: MSE for Multiple Classes

- We still use augmented feature vectors $\boldsymbol{y_1},\ldots, \boldsymbol{y_n}$
  - but do not multiply by -1
- Define ***m*** linear discriminant functions

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

- Given ***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 ***i***, makes sense to seek weight vector $\boldsymbol{a_i}$, s.t.

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

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

## LDF: MSE for Multiple Classes

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

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

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

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{bmatrix} = \begin{bmatrix} \text{sample from class } 1 \\ \text{sample from class } 1 \\ \vdots \\ \text{sample from class } m \\ \text{sample from class } m \end{bmatrix}$$

## 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 $\boldsymbol{1}$:

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

- We need to solve: $Ya_i = b_i$

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

## LDF: MSE for Multiple Classes

- We need to solve $\boldsymbol{Y a_i} = \boldsymbol{b_i}$
- Usually no exact solution since $\boldsymbol{Y}$ is overdetermined
- Use least squares to minimize norm of the error vector $\| \boldsymbol{Y a_i} - \boldsymbol{b_i} \|$
- 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 $\boldsymbol{m}$ LSE problems, one for each class
- Can write these $\boldsymbol{m}$ 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 $\boldsymbol{d+1}$ 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 $\boldsymbol{YA} = \boldsymbol{B}$:

$$\underbrace{\begin{bmatrix} \text{sample from class1} \\ \text{sample from class1} \\ \text{sample from class2} \\ \text{sample from class3} \\ \text{sample from class3} \\ \text{sample from class3} \end{bmatrix}}_{\boldsymbol{Y}} \underbrace{\begin{bmatrix} \text{weights for c1} & \text{weights for c2} & \text{weights for c3} \end{bmatrix}}_{\boldsymbol{A}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{B}}$$

## LDF: MSE for Multiple Classes

- Our objective function is:

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

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

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

## 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