

CS4442/9542b
Artificial Intelligence II
prof. Olga Veksler

Lecture 4
Machine Learning
Linear Classifier

Outline

- Optimization with gradient descent
- Linear Classifier
 - Two classes
 - Multiple classes
 - Perceptron Criterion Function
 - Batch perceptron rule
 - Single sample perceptron rule
 - Minimum Squared Error (MSE) rule
 - Pseudoinverse
 - Gradient descent (Widrow-Hoff Procedure)

Optimization

- How to minimize a function of a single variable

$$J(\mathbf{x}) = (\mathbf{x} - 5)^2$$

- From calculus, take derivative, set it to 0

$$\frac{d}{d\mathbf{x}} J(\mathbf{x}) = 0$$

- Solve the resulting equation
 - maybe easy or hard to solve

- Example above is easy:

$$\frac{d}{d\mathbf{x}} J(\mathbf{x}) = 2(\mathbf{x} - 5) = 0 \Rightarrow \mathbf{x} = 5$$

Optimization

- How to minimize a function of many variables

$$\mathbf{J}(\mathbf{x}) = \mathbf{J}(\mathbf{x}_1, \dots, \mathbf{x}_d)$$

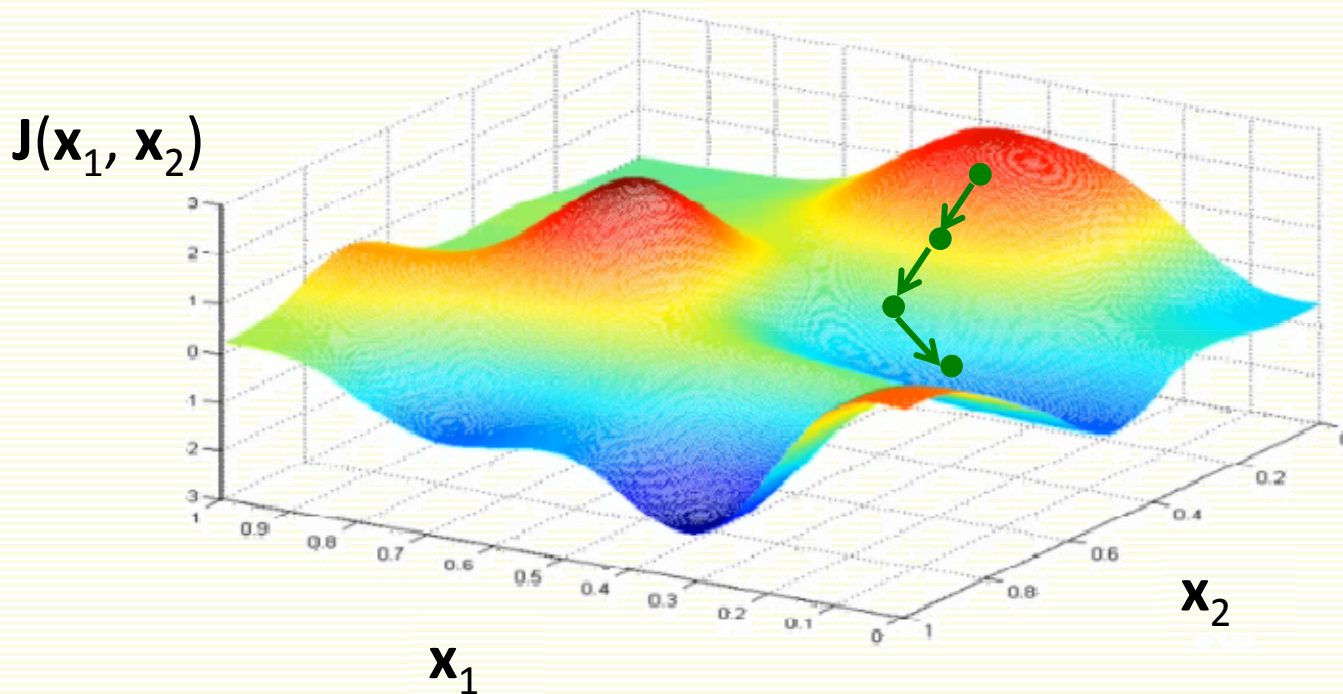
- From calculus, take partial derivatives, set them to 0

gradient

$$\begin{bmatrix} \frac{\partial}{\partial \mathbf{x}_1} \mathbf{J}(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial \mathbf{x}_d} \mathbf{J}(\mathbf{x}) \end{bmatrix} = \nabla \mathbf{J}(\mathbf{x}) = \mathbf{0}$$

- Solve the resulting system of \mathbf{d} equations
- It may not be possible to solve the system of equations above analytically

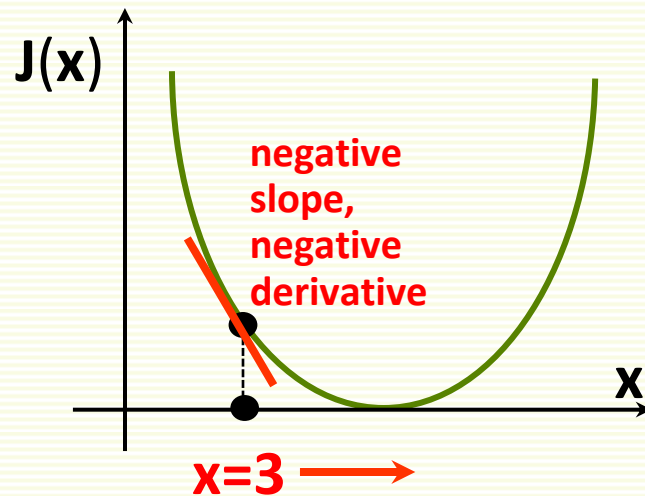
Optimization: Gradient Direction



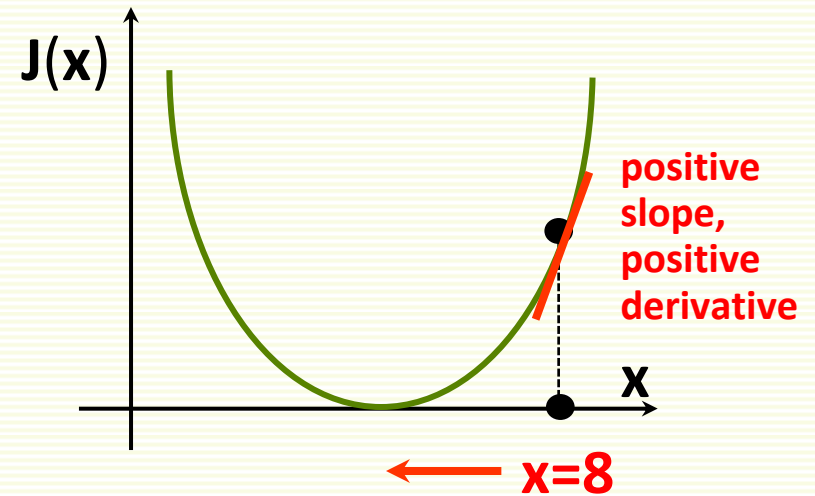
- Gradient $\nabla J(\mathbf{x})$ points in the direction of steepest increase of function $J(\mathbf{x})$
- $-\nabla J(\mathbf{x})$ points in the direction of steepest decrease

Gradient Direction in 1D

- Gradient is just derivative in 1D
- Example: $J(x) = (x-5)^2$ and derivative is $\frac{d}{dx}J(x) = 2(x-5)$



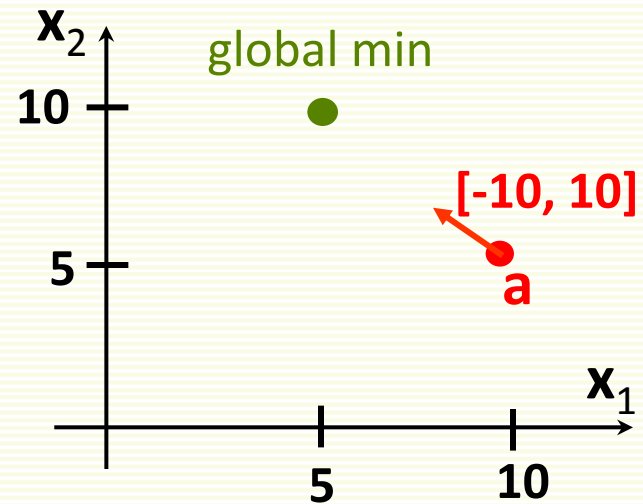
- Let $x = 3$
- $-\frac{d}{dx}J(3) = 4$
- derivative says increase x



- Let $x = 8$
- $-\frac{d}{dx}J(8) = -6$
- derivative says decrease x

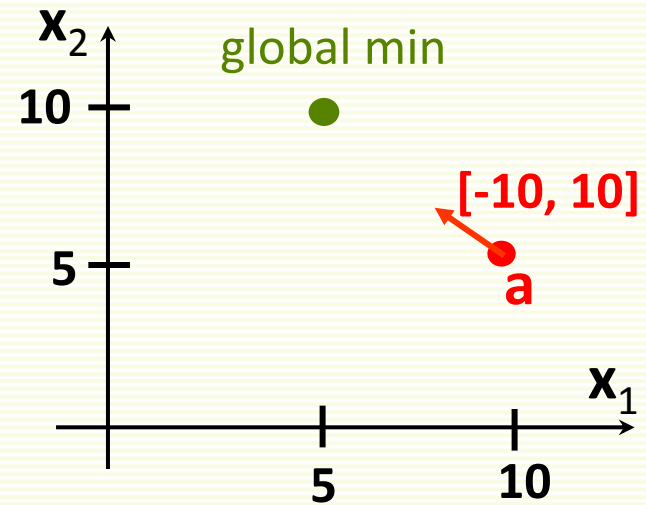
Gradient Direction in 2D

- $J(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - 5)^2 + (\mathbf{x}_2 - 10)^2$
- $\frac{\partial}{\partial \mathbf{x}_1} J(\mathbf{x}) = 2(\mathbf{x}_1 - 5)$
- $\frac{\partial}{\partial \mathbf{x}_2} J(\mathbf{x}) = 2(\mathbf{x}_2 - 10)$
- Let $\mathbf{a} = [10, 5]$
- $-\frac{\partial}{\partial \mathbf{x}_1} J(\mathbf{a}) = -10$
- $-\frac{\partial}{\partial \mathbf{x}_2} J(\mathbf{a}) = 10$



Gradient Descent: Step Size

- $J(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - 5)^2 + (\mathbf{x}_2 - 10)^2$
- Which step size to take?
- Controlled by parameter α
 - called **learning rate**
- From previous example:
 - $\mathbf{a} = [10 \ 5]$
 - $-\nabla J(\mathbf{a}) = [-10 \ 10]$
- Let $\alpha = 0.2$
- $\mathbf{a} - \alpha \nabla J(\mathbf{a}) = [10 \ 5] + 0.2 [-10 \ 10] = [8 \ 7]$
- $J(10, 5) = 50$; $J(8, 7) = 18$



Gradient Descent Algorithm

$k = 1$

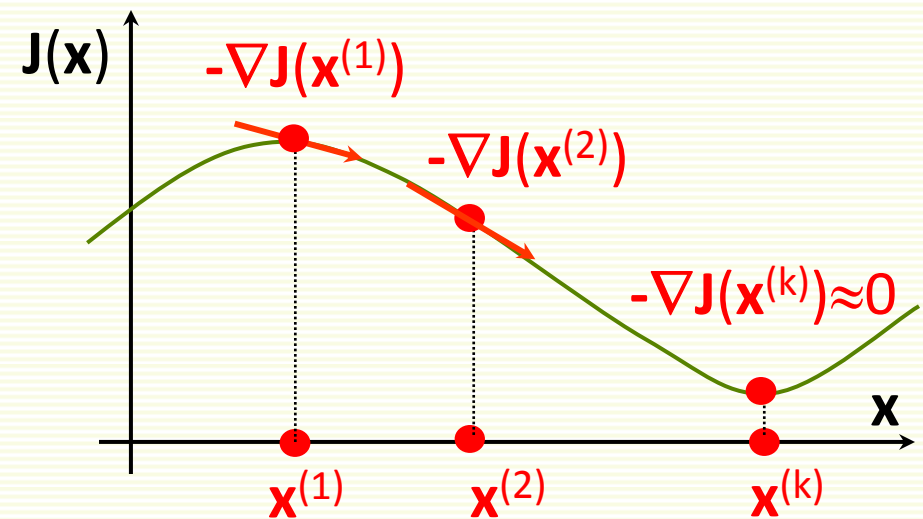
$\mathbf{x}^{(1)}$ = any initial guess

choose α, ϵ

while $\alpha \|\nabla J(\mathbf{x}^{(k)})\| > \epsilon$

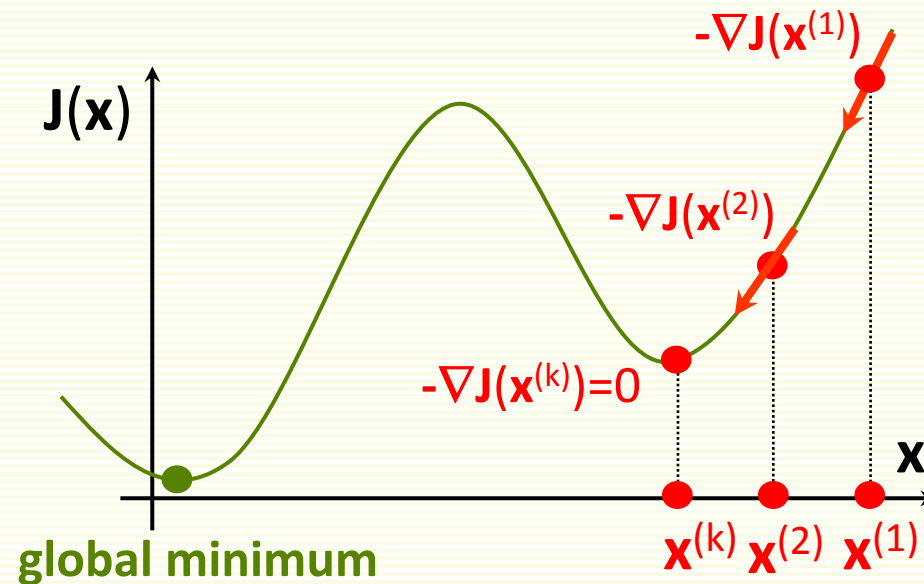
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla J(\mathbf{x}^{(k)})$$

$k = k + 1$



Gradient Descent: Local Minimum

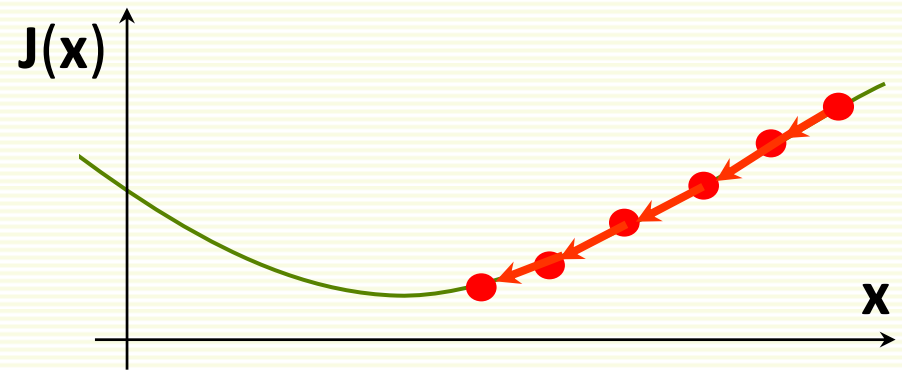
- Not guaranteed to find global minimum
 - gets stuck in local minimum



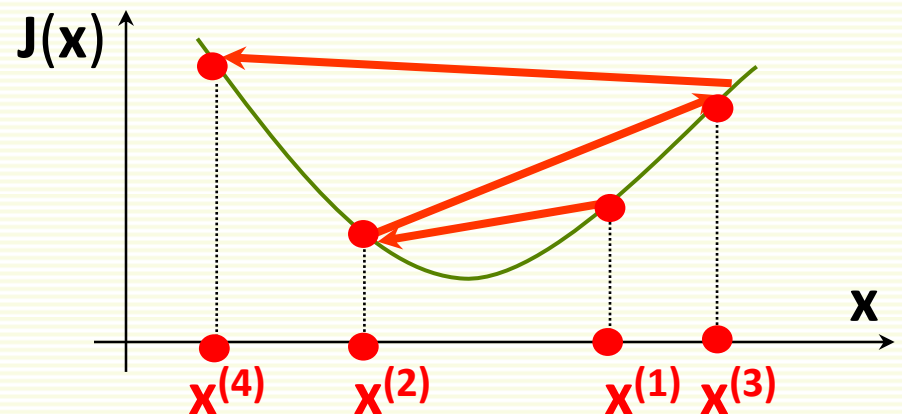
- Still gradient descent is very popular because it is simple and applicable to any differentiable function

How to Set Learning Rate α ?

- If α too small, too many iterations to converge



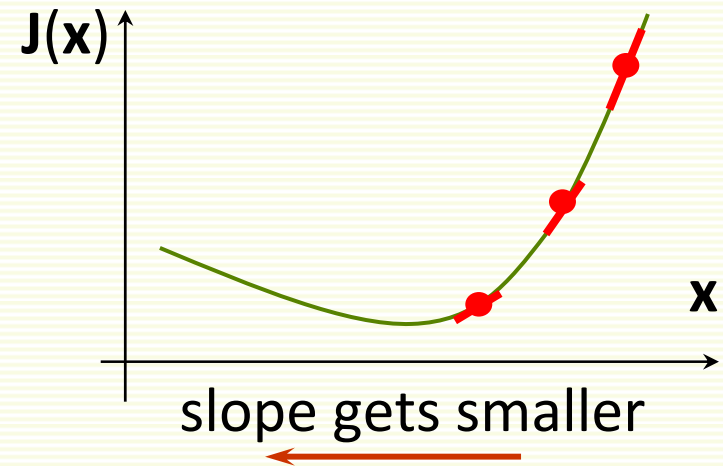
- If α too large, may overshoot the local minimum and possibly never even converge



- It helps to compute $J(x)$ as a function of iteration number, to make sure we are properly minimizing it

How to Set Learning Rate α ?

- As we approach local minimum, often gradient gets smaller
- Step size may get smaller automatically, even if α is fixed
- So it may be unnecessary to decrease α over time in order not to overshoot a local minimum



Variable Learning Rate

- If desired, can change learning rate α at each iteration

k = 1

$\mathbf{x}^{(1)}$ = any initial guess

choose α, ε

while $\alpha \|\nabla J(\mathbf{x}^{(k)})\| > \varepsilon$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla J(\mathbf{x}^{(k)})$$

k = k + 1



k = 1

$\mathbf{x}^{(1)}$ = any initial guess

choose ε

while $\alpha \|\nabla J(\mathbf{x}^{(k)})\| > \varepsilon$

choose $\alpha^{(k)}$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha^{(k)} \nabla J(\mathbf{x}^{(k)})$$

k = k + 1

Variable Learning Rate

- Usually don't keep track of all intermediate solutions

$k = 1$

$\mathbf{x}^{(1)}$ = any initial guess

choose α, ε

while $\alpha \|\nabla J(\mathbf{x}^{(k)})\| > \varepsilon$

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla J(\mathbf{x}^{(k)})$

$k = k + 1$



\mathbf{x} = any initial guess

choose α, ε

while $\alpha \|\nabla J(\mathbf{x})\| > \varepsilon$

$\mathbf{x} = \mathbf{x} - \alpha \nabla J(\mathbf{x})$

Advanced Optimization Methods

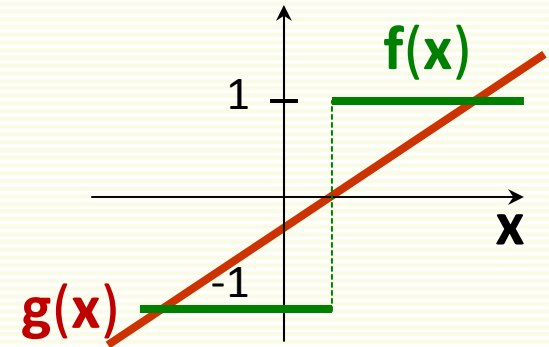
- There are more advanced gradient-based optimization methods
- Such as conjugate gradient
 - automatically pick a good learning rate α
 - usually converge faster
 - however more complex to understand and implement
 - in Matlab, use **fminunc** for various advanced optimization methods

Supervised Machine Learning (Recap)

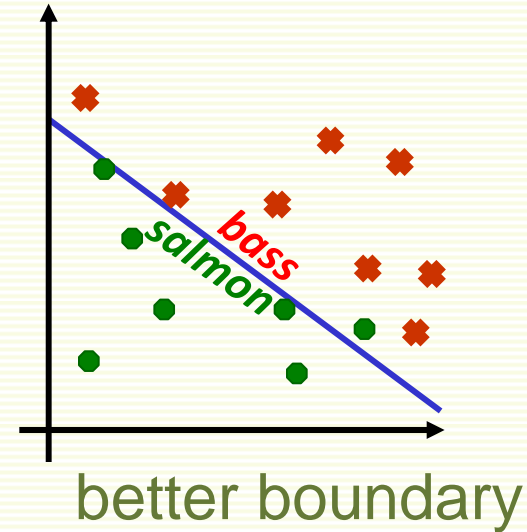
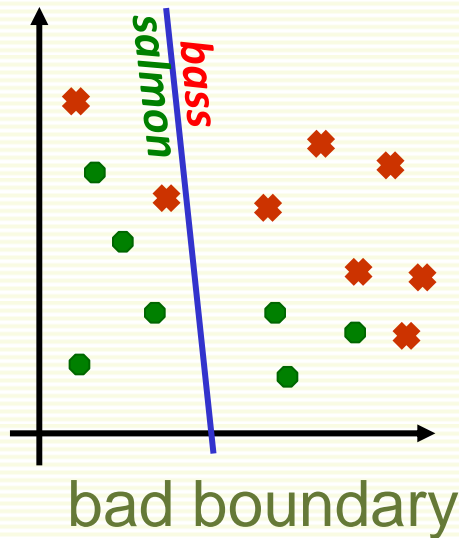
- Chose a *learning machine* $f(\mathbf{x}, \mathbf{w})$
 - \mathbf{w} are tunable weights, \mathbf{x} is the input example
 - $f(\mathbf{x}, \mathbf{w})$ should output the correct class of sample \mathbf{x}
 - use labeled samples to tune weights \mathbf{w} so that $f(\mathbf{x}, \mathbf{w})$ give the correct class (correct \mathbf{y}) for example \mathbf{x}
- How to choose a learning machine $f(\mathbf{x}, \mathbf{w})$?
 - many choices possible
 - previous lecture: kNN classifier
 - this lecture: linear classifier

Linear Classifier: 2 Classes

- First consider the two-class case
- We choose the following encoding:
 - $y = 1$ for the first class
 - $y = -1$ for the second class
- Linear classifier
 - linear function: $-\infty \leq \mathbf{w}_0 + \mathbf{x}_1 \mathbf{w}_1 + \dots + \mathbf{x}_d \mathbf{w}_d \leq \infty$
 - we need $\mathbf{f}(\mathbf{x}, \mathbf{w})$ to be either $+1$ or -1
 - let $\mathbf{g}(\mathbf{x}, \mathbf{w}) = \mathbf{w}_0 + \mathbf{x}_1 \mathbf{w}_1 + \dots + \mathbf{x}_d \mathbf{w}_d$
 - let $\mathbf{f}(\mathbf{x}, \mathbf{w}) = \text{sign}(\mathbf{g}(\mathbf{x}, \mathbf{w}))$
 - 1 if $\mathbf{g}(\mathbf{x}, \mathbf{w})$ is positive
 - -1 if $\mathbf{g}(\mathbf{x}, \mathbf{w})$ is negative
 - $\mathbf{g}(\mathbf{x}, \mathbf{w})$ is called the *discriminant function*



Linear Classifier: Decision Boundary



- $f(\mathbf{x}, \mathbf{w}) = \text{sign}(\mathbf{g}(\mathbf{x}, \mathbf{w})) = \text{sign}(\mathbf{w}_0 + \mathbf{x}_1 \mathbf{w}_1 + \dots + \mathbf{x}_d \mathbf{w}_d)$
- Decision boundary is linear
- Find the best linear boundary to separate two classes
- Search for best $\mathbf{w} = [\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_d]$ to minimize training error

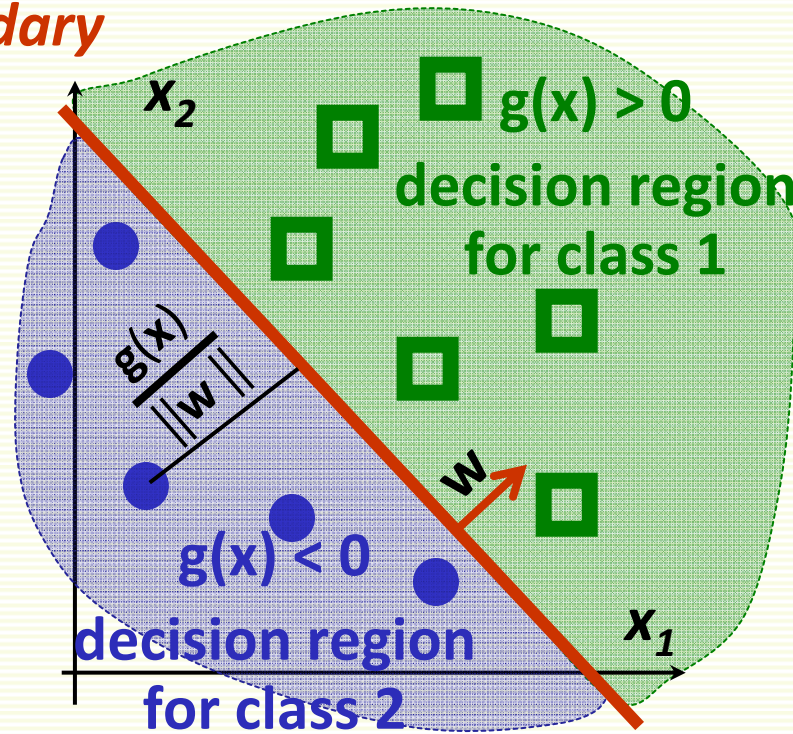
More on Linear Discriminant Function (LDF)

- LDF: $g(\mathbf{x}, \mathbf{w}) = \mathbf{w}_0 + \mathbf{x}_1 \mathbf{w}_1 + \dots + \mathbf{x}_d \mathbf{w}_d$

- Written using vector notation $g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + \mathbf{w}_0$
weight vector bias or threshold

decision boundary

$g(\mathbf{x}) = 0$

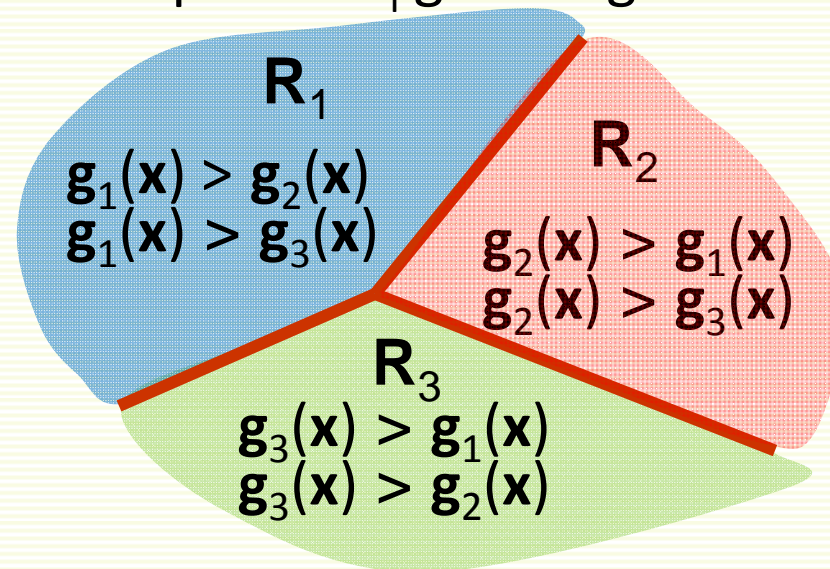


More on Linear Discriminant Function (LDF)

- Decision boundary: $\mathbf{g}(\mathbf{x}, \mathbf{w}) = \mathbf{w}_0 + \mathbf{x}_1 \mathbf{w}_1 + \dots + \mathbf{x}_d \mathbf{w}_d = 0$
- This is a hyperplane, by definition
 - a point in 1D
 - a line in 2D
 - a plane in 3D
 - a hyperplane in higher dimensions

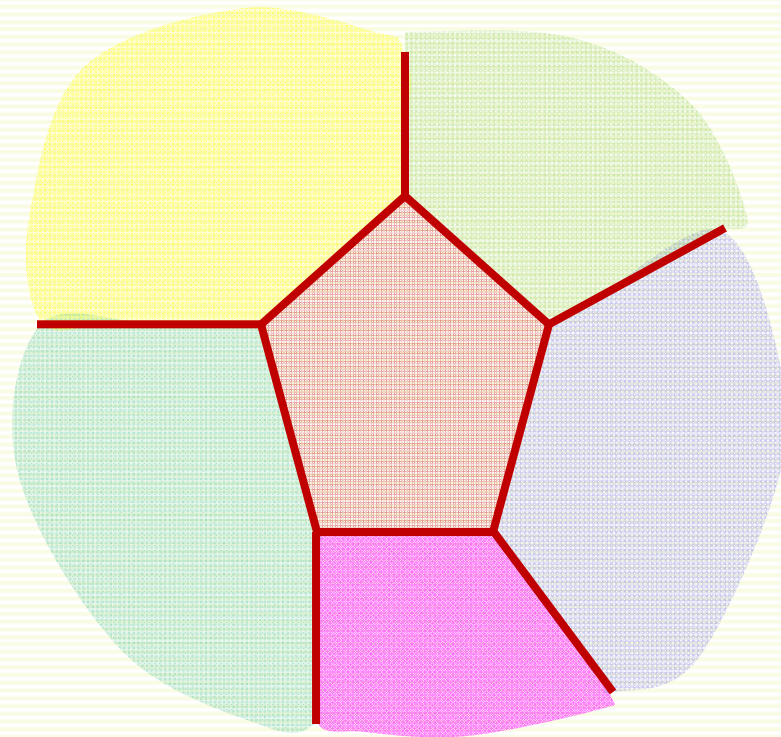
Multiple Classes

- We have m classes
- Define m linear discriminant functions
$$\mathbf{g}_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + \mathbf{w}_{i0} \text{ for } i = 1, 2, \dots, m$$
- Assign \mathbf{x} to class i if
$$\mathbf{g}_i(\mathbf{x}) > \mathbf{g}_j(\mathbf{x}) \text{ for all } j \neq i$$
- Let \mathbf{R}_i be the decision region for class i
 - That is all examples in \mathbf{R}_i get assigned class i



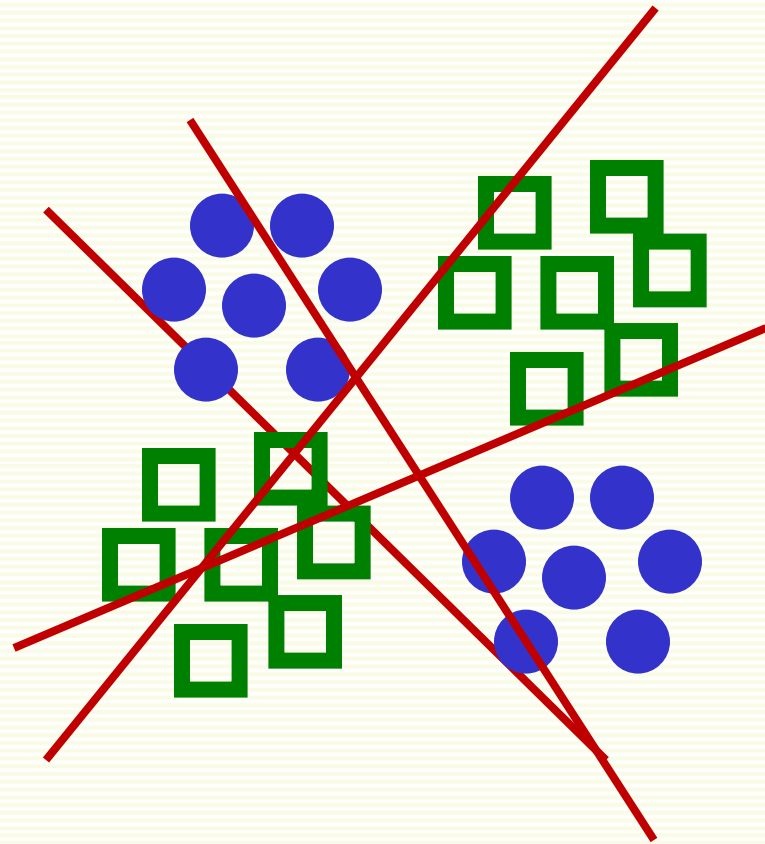
Multiple Classes

- Can be shown that decision regions are convex
- In particular, they must be spatially contiguous



Failure Cases for Linear Classifier

- Thus applicability of linear classifiers is limited to mostly unimodal distributions, such as Gaussian
- Not unimodal data
- Need non-contiguous decision regions
- Linear classifier will fail



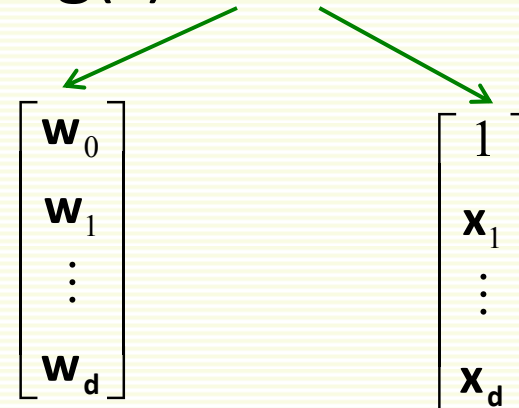
Fitting Parameters w

- Linear discriminant function $g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$

- Can rewrite it $g(\mathbf{x}) = \underbrace{[w_0 \quad \mathbf{w}^t]}_{\substack{\text{new weight} \\ \text{vector } \mathbf{a}}} \underbrace{\begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}}_{\substack{\text{new} \\ \text{feature} \\ \text{vector } \mathbf{z}}} = \mathbf{a}^t \mathbf{z} = g(\mathbf{z})$

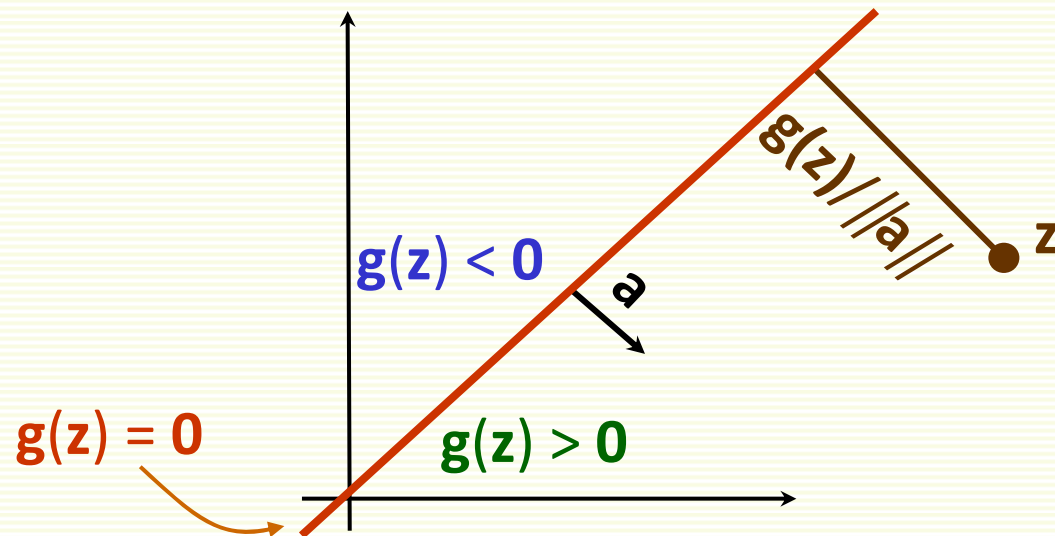
- \mathbf{z} is called augmented feature vector

- new problem equivalent to the old $g(\mathbf{z}) = \mathbf{a}^t \mathbf{z}$



Augmented Feature Vector

- Feature augmenting is done to simplify notation
- From now on we assume that we have augmented feature vectors
 - given samples $\mathbf{x}^1, \dots, \mathbf{x}^n$ convert them to augmented samples $\mathbf{z}^1, \dots, \mathbf{z}^n$ by adding a new dimension of value 1
- $g(\mathbf{z}) = \mathbf{a}^t \mathbf{z}$



Training Error

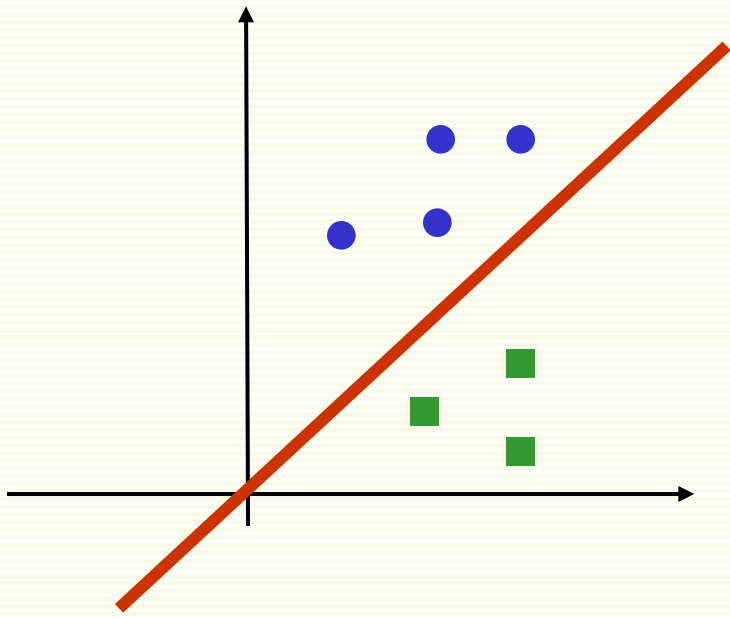
- For the rest of the lecture, assume we have 2 classes
- Samples $\mathbf{z}^1, \dots, \mathbf{z}^n$ some in class 1, some in class 2
- Use these samples to determine weights \mathbf{a} in the discriminant function $\mathbf{g}(\mathbf{z}) = \mathbf{a}^t \mathbf{z}$
- Want to minimize number of misclassified samples
- Recall that
$$\begin{cases} \mathbf{g}(\mathbf{z}^i) > 0 & \Rightarrow \text{class 1} \\ \mathbf{g}(\mathbf{z}^i) < 0 & \Rightarrow \text{class 2} \end{cases}$$
- Thus training error is 0 if
$$\begin{cases} \mathbf{g}(\mathbf{z}^i) \geq 0 & \forall \mathbf{z}^i \text{ class 1} \\ \mathbf{g}(\mathbf{z}^i) < 0 & \forall \mathbf{z}^i \text{ class 2} \end{cases}$$

Simplifying Notation Further

- Thus training error is 0 if
$$\begin{cases} \mathbf{a}^t \mathbf{z}^i > 0 & \forall \mathbf{z}^i \text{ class 1} \\ \mathbf{a}^t \mathbf{z}^i < 0 & \forall \mathbf{z}^i \text{ class 2} \end{cases}$$
- Equivalently, training error is 0 if
$$\begin{cases} \mathbf{a}^t \mathbf{z}^i > 0 & \forall \mathbf{z}^i \text{ class 1} \\ \mathbf{a}^t (-\mathbf{z}^i) > 0 & \forall \mathbf{z}^i \text{ class 2} \end{cases}$$
- Problem “normalization”:
 1. replace all examples \mathbf{z}^i from class 2 by $-\mathbf{z}^i$
 2. seek weights \mathbf{a} s.t. $\mathbf{a}^t \mathbf{z}^i > 0$ for $\forall \mathbf{z}^i$
- If exists, such \mathbf{a} is called a ***separating*** or ***solution*** vector
- Original samples $\mathbf{x}^1, \dots, \mathbf{x}^n$ can also be linearly separated

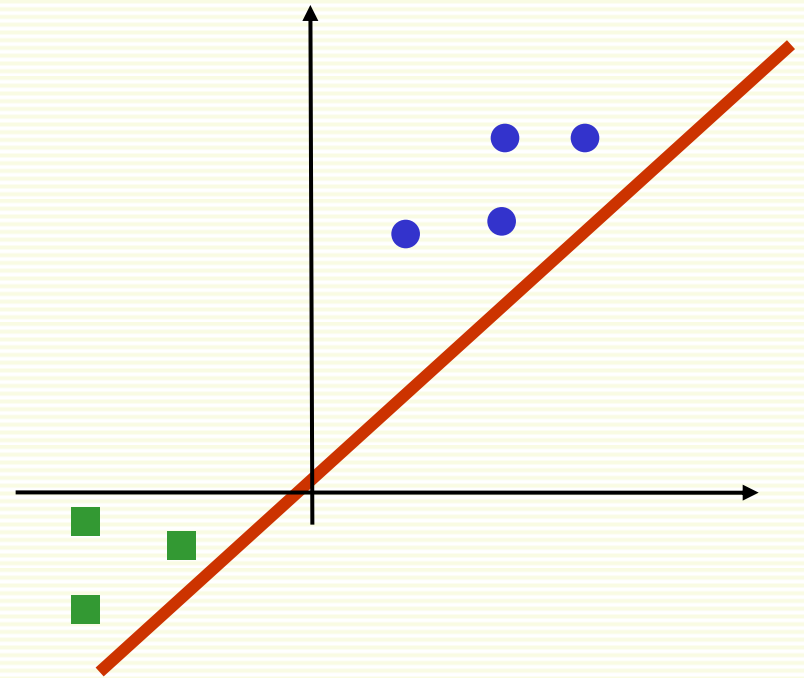
Effect of Normalization

before normalization



seek a hyperplane that separates samples from different categories

after normalization

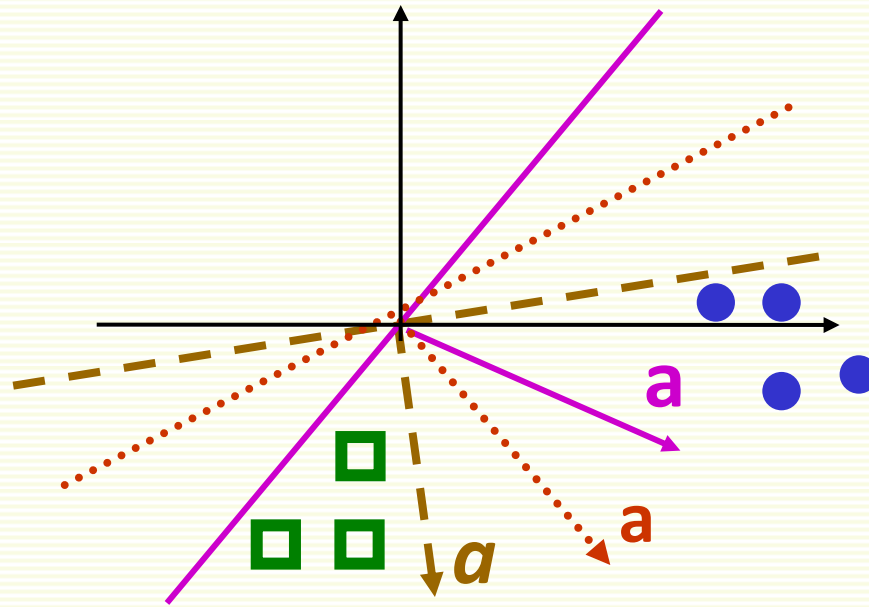


seek hyperplane that puts **normalized** samples on the same (positive) side

Solution Region

- Find weight vector \mathbf{a} s.t. for all samples $\mathbf{z}^1, \dots, \mathbf{z}^n$

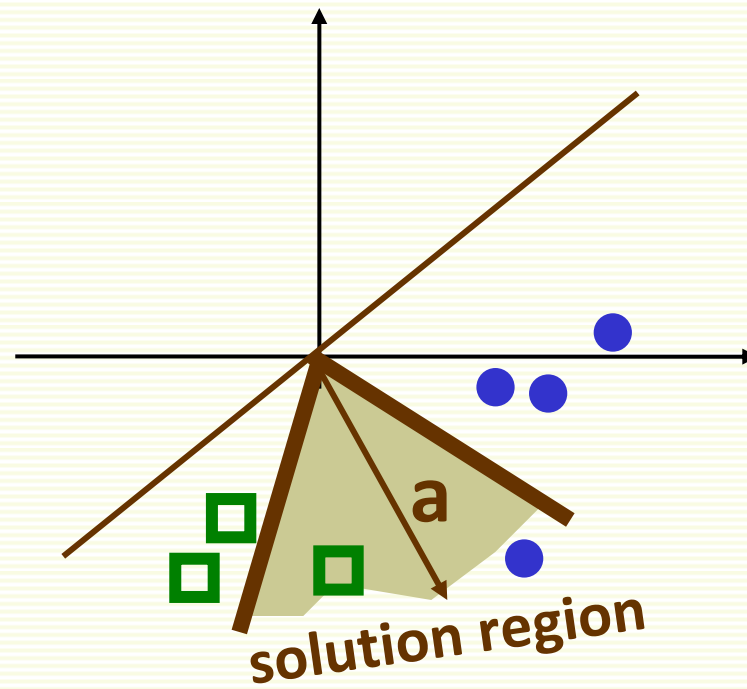
$$\mathbf{a}^t \mathbf{z}^i = \sum_{k=0}^d \mathbf{a}_k \mathbf{z}_k^i > 0$$



- If there is one such \mathbf{a} , then there are infinitely many \mathbf{a}

Solution Region

- Solution region: the set of all possible solutions for a



Criterion Function: First Attempt

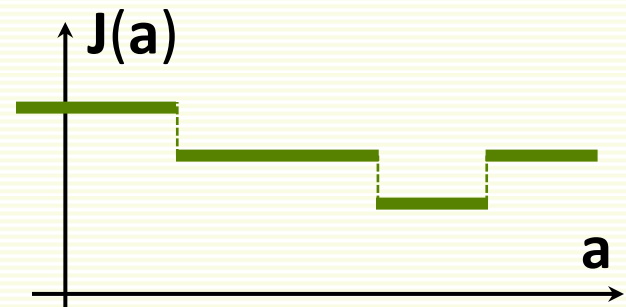
- Find weight vector \mathbf{a} s.t. $\forall \mathbf{z}^1, \dots, \mathbf{z}^n, \mathbf{a}^t \mathbf{z}^i > 0$
- Design a criterion function $J(\mathbf{a})$, which is minimum when \mathbf{a} is a solution vector
- Let $Z(\mathbf{a})$ be the set of examples misclassified by \mathbf{a}

$$Z(\mathbf{a}) = \{ \mathbf{z}^i \mid \mathbf{a}^t \mathbf{z}^i < 0 \}$$

- Natural choice: number of misclassified examples

$$J(\mathbf{a}) = |Z(\mathbf{a})|$$

- Unfortunately, can't be minimized with gradient descent
 - piecewise constant, gradient zero or does not exist

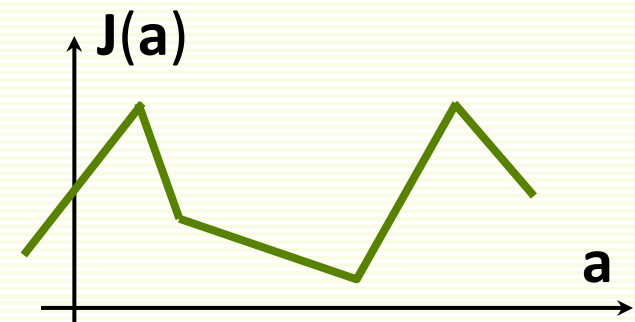
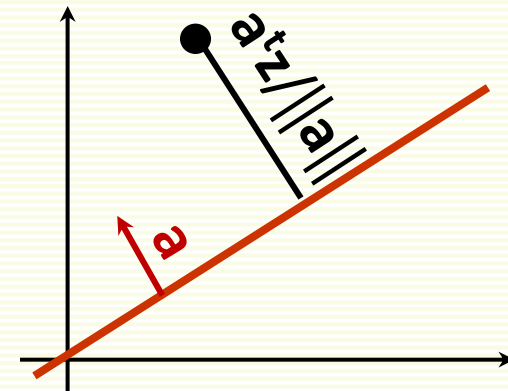


Perceptron Criterion Function

- Better choice: Perceptron criterion function

$$J_p(\mathbf{a}) = \sum_{z \in Z(\mathbf{a})} (-\mathbf{a}^t \mathbf{z})$$

- If \mathbf{z} is misclassified, $\mathbf{a}^t \mathbf{z} < 0$
- Thus $J(\mathbf{a}) \geq 0$
- $J_p(\mathbf{a})$ is proportional to the sum of distances of misclassified examples to decision boundary
- $J_p(\mathbf{a})$ is piecewise linear and suitable for gradient descent



Optimizing with Gradient Descent

$$J_p(\mathbf{a}) = \sum_{\mathbf{z} \in \mathbf{Z}(\mathbf{a})} (-\mathbf{a}^t \mathbf{z})$$

- Gradient of $J_p(\mathbf{a})$ is $\nabla J_p(\mathbf{a}) = \sum_{\mathbf{z} \in \mathbf{Z}(\mathbf{a})} (-\mathbf{z})$
 - cannot solve $\nabla J_p(\mathbf{a}) = 0$ analytically because of $\mathbf{Z}(\mathbf{a})$

- Recall update rule for gradient descent

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla J(\mathbf{x}^{(k)})$$

- Gradient decent update rule for $J_p(\mathbf{a})$ is:

$$\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \alpha \sum_{\mathbf{z} \in \mathbf{Z}(\mathbf{a})} \mathbf{z}$$

- called **batch rule** because it is based on all examples
- true gradient descent

Perceptron Single Sample Rule

- Gradient decent single sample rule for $J_p(\mathbf{a})$ is

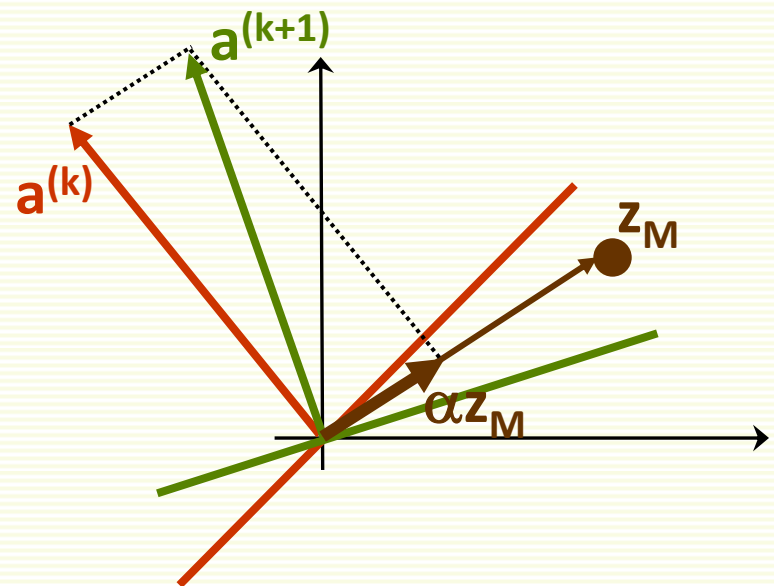
$$\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \alpha \cdot \mathbf{z}_M$$

- \mathbf{z}_M is one sample misclassified by $\mathbf{a}^{(k)}$
 - must have a consistent way to visit samples
- Geometric Interpretation:

- \mathbf{z}_M misclassified by $\mathbf{a}^{(k)}$

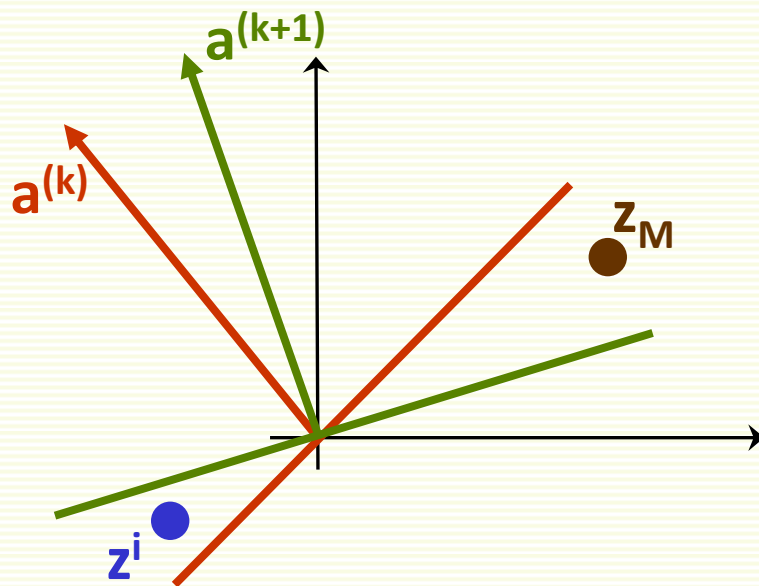
$$(\mathbf{a}^{(k)})^t \mathbf{z}_M \leq 0$$

- \mathbf{z}_M is on the wrong side of decision boundary
- adding $\alpha \cdot \mathbf{z}_M$ to \mathbf{a} moves decision boundary in the right direction

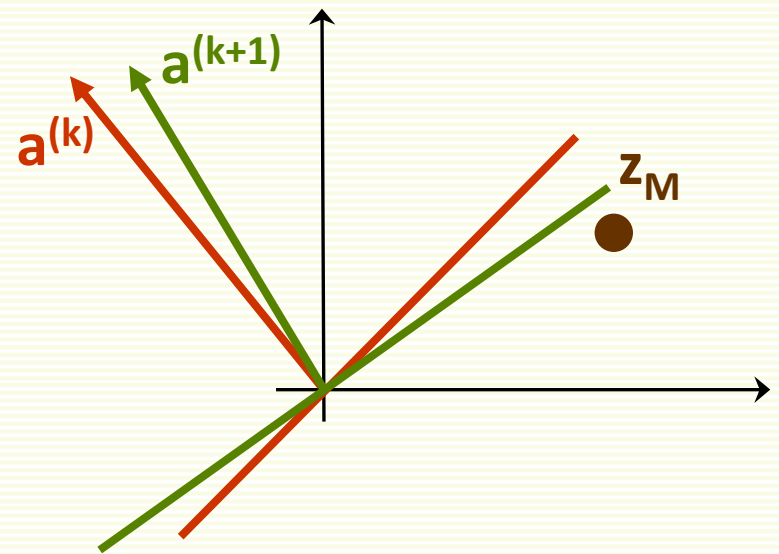


Perceptron Single Sample Rule

if α is too large, previously correctly classified sample \mathbf{z}^i is now misclassified



if α is too small, \mathbf{z}_M is still misclassified



Perceptron Single Sample Rule Example

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

- class 1: students who get grade A
- class 2: students who get grade F

Augment Feature Vector

	features					grade
<i>name</i>	<i>extra</i>	<i>good attendance?</i>	<i>tall?</i>	<i>sleeps in class?</i>	<i>chews gum?</i>	
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 $\mathbf{x}^1, \dots, \mathbf{x}^n$ to augmented samples $\mathbf{z}^1, \dots, \mathbf{z}^n$ by adding a new dimension of value 1

“Normalization”

	features					grade
<i>name</i>	<i>extra</i>	<i>good attendance?</i>	<i>tall?</i>	<i>sleeps in class?</i>	<i>chews gum?</i>	
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 2 by their negative

$$\mathbf{z}^i \rightarrow -\mathbf{z}^i$$

- Seek weight vector \mathbf{a} s.t. $\mathbf{a}^t \mathbf{z}^i > 0$ for all \mathbf{z}^i

Apply Single Sample Rule

	features					grade
<i>name</i>	<i>extra</i>	<i>good attendance?</i>	<i>tall?</i>	<i>sleeps in class?</i>	<i>chews gum?</i>	
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

- Gradient descent single sample rule: $\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \alpha \cdot \mathbf{z}_M$
- Set fixed learning rate to $\alpha = 1$: $\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \mathbf{z}_M$
- Sample is misclassified if $\mathbf{a}^t \mathbf{z}^i = \sum_{k=0}^4 \mathbf{a}_k \mathbf{z}_k^i < 0$

Apply Single Sample Rule

- initial weights $\mathbf{a}^{(1)} = [0.25, 0.25, 0.25, 0.25]$
- visit all samples sequentially

<i>name</i>	$\mathbf{a}^t \mathbf{z}$	<i>misclassified?</i>
Jane	$0.25*1+0.25*1+0.25*1+0.25*(-1)+0.25*(-1) > 0$	<i>no</i>
Steve	$0.25*(-1)+0.25*(-1)+0.25*(-1)+0.25*(-1)+0.25*(-1) < 0$	<i>yes</i>

- new weights

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

Apply Single Sample Rule

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

<i>name</i>	$\mathbf{a}^t \mathbf{z}$	<i>misclassified?</i>
Mary	$-0.75 * (-1) - 0.75 * 1 - 0.75 * 1 - 0.75 * 1 - 0.75 * (-1) < 0$	yes

- new weights

$$\begin{aligned} \mathbf{a}^{(3)} &= \mathbf{a}^{(2)} + \mathbf{z}_M = [-0.75 \quad -0.75 \quad -0.75 \quad -0.75 \quad -0.75] + \\ &\quad + [-1 \quad 1 \quad 1 \quad 1 \quad -1] = \\ &= [-1.75 \quad 0.25 \quad 0.25 \quad 0.25 \quad -1.75] \end{aligned}$$

Apply Single Sample Rule

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

<i>name</i>	$\mathbf{a}^t \mathbf{z}$	<i>misclassified?</i>
Peter	$-1.75 * 1 + 0.25 * 1 + 0.25 * (-1) + 0.25 * (-1) - 1.75 * 1 < 0$	yes

- new weights

$$\begin{aligned} \mathbf{a}^{(4)} &= \mathbf{a}^{(3)} + \mathbf{z}_M = [-1.75 \quad 0.25 \quad 0.25 \quad 0.25 \quad -1.75] + \\ &\quad + [1 \quad 1 \quad -1 \quad -1 \quad 1] = \\ &= [-0.75 \quad 1.25 \quad -0.75 \quad -0.75 \quad -0.75] \end{aligned}$$

Single Sample Rule: Convergence

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

<i>name</i>	$\mathbf{a}^t \mathbf{z}$	<i>misclassified?</i>
Jane	$-0.75 * 1 + 1.25 * 1 - 0.75 * 1 - 0.75 * (-1) - 0.75 * (-1) + 0$	<i>no</i>
Steve	$-0.75 * (-1) + 1.25 * (-1) - 0.75 * (-1) - 0.75 * (-1) - 0.75 * (-1) > 0$	<i>no</i>
Mary	$-0.75 * (-1) + 1.25 * 1 - 0.75 * 1 - 0.75 * 1 - 0.75 * (-1) > 0$	<i>no</i>
Peter	$-0.75 * 1 + 1.25 * 1 - 0.75 * (-1) - 0.75 * (-1) - 0.75 * 1 > 0$	<i>no</i>

- Thus the discriminant function is

$$\mathbf{g}(\mathbf{z}) = -0.75 \mathbf{z}_0 + 1.25 \mathbf{z}_1 - 0.75 \mathbf{z}_2 - 0.75 \mathbf{z}_3 - 0.75 \mathbf{z}_4$$

- Converting back to the original features \mathbf{x}

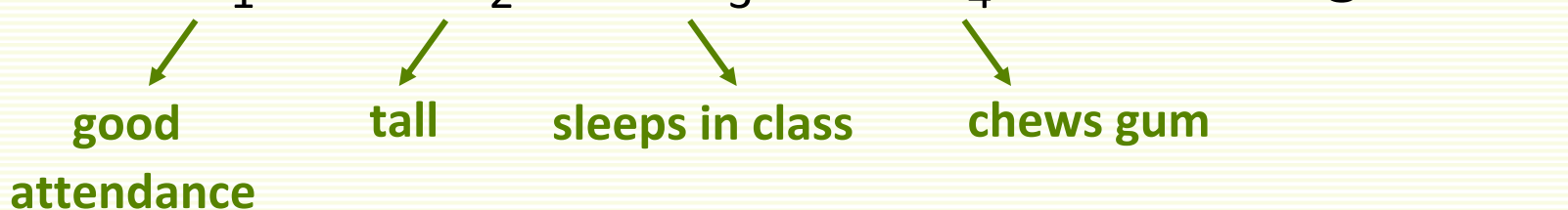
$$\mathbf{g}(\mathbf{x}) = 1.25 \mathbf{x}_1 - 0.75 \mathbf{x}_2 - 0.75 \mathbf{x}_3 - 0.75 \mathbf{x}_4 - 0.75$$

Final Classifier

- Trained LDF: $\mathbf{g}(\mathbf{x}) = 1.25x_1 - 0.75x_2 - 0.75x_3 - 0.75x_4 - 0.75$

- Leads to classifier:

$$1.25x_1 - 0.75x_2 - 0.75x_3 - 0.75x_4 > 0.75 \Rightarrow \text{grade A}$$



good attendance tall sleeps in class chews gum

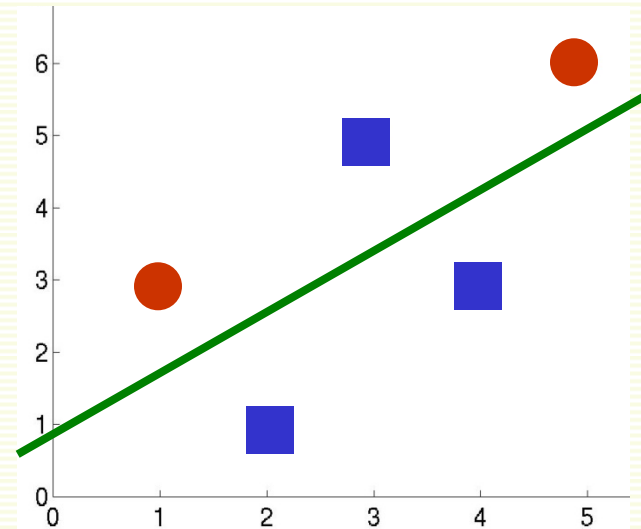
- This is just *one* possible solution vector
- With $\mathbf{a}^{(1)} = [0, 0.5, 0.5, 0, 0]$, solution is $[-1, 1.5, -0.5, -1, -1]$

$$1.5x_1 - 0.5x_2 - x_3 - x_4 > 1 \Rightarrow \text{grade A}$$

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

Non-Linearly Separable Case

- Suppose we have examples:
 - class 1: [2,1], [4,3], [3,5]
 - class 2: [1,3], [5,6]
 - not linearly separable
- Still would like to get approximate separation
- Good line choice is shown in green
- Let us run gradient descent
 - Add extra feature and “normalize”



$$\mathbf{z}^1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

$$\mathbf{z}^2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}$$

$$\mathbf{z}^3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$$

$$\mathbf{z}^4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix}$$

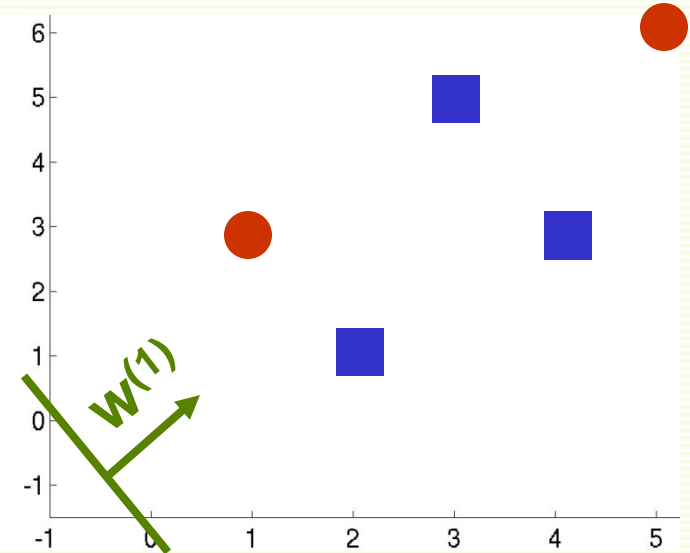
$$\mathbf{z}^5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$

Non-Linearly Separable Case

- single sample perceptron rule
- Initial weights $\mathbf{a}^{(1)} = [1 \ 1 \ 1]$
- This is line $\mathbf{x}_1 + \mathbf{x}_2 + 1 = 0$
- Use fixed learning rate $\alpha = 1$
- Rule is: $\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \mathbf{z}_M$

$$\mathbf{z}^1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \mathbf{z}^2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad \mathbf{z}^3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad \mathbf{z}^4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad \mathbf{z}^5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$

- $\mathbf{a}^t \mathbf{z}^1 = [1 \ 1 \ 1] \cdot [1 \ 2 \ 1]^t > 0$
- $\mathbf{a}^t \mathbf{z}^2 = [1 \ 1 \ 1] \cdot [1 \ 4 \ 3]^t > 0$
- $\mathbf{a}^t \mathbf{z}^3 = [1 \ 1 \ 1] \cdot [1 \ 3 \ 5]^t > 0$



Non-Linearly Separable Case

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

- rule is: $\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \mathbf{z}_M$

$$\mathbf{z}^1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \mathbf{z}^2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad \mathbf{z}^3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad \mathbf{z}^4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad \mathbf{z}^5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$

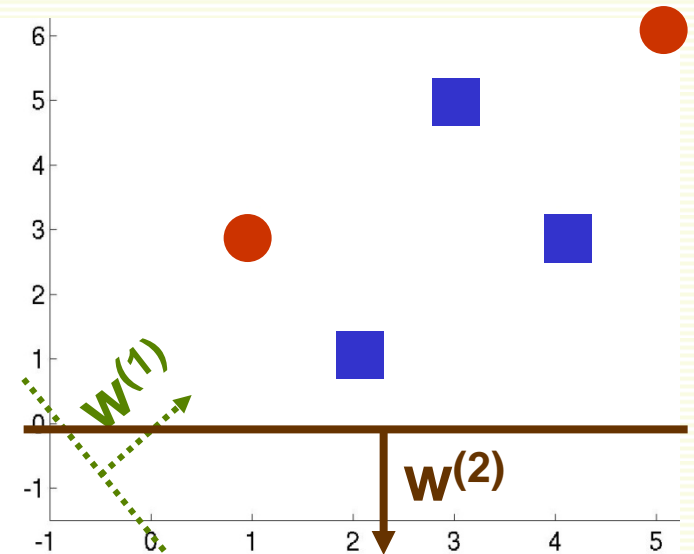
- $\mathbf{a}^t \mathbf{z}^4 = [1 \ 1 \ 1] \cdot [-1 \ -1 \ -3]^t = -5 < 0$

- *Update:* $\mathbf{a}^{(2)} = \mathbf{a}^{(1)} + \mathbf{z}_M = [1 \ 1 \ 1] + [-1 \ -1 \ -3] = [0 \ 0 \ -2]$

- $\mathbf{a}^t \mathbf{z}^5 = [0 \ 0 \ -2] \cdot [-1 \ -5 \ -6]^t = 12 > 0$

- $\mathbf{a}^t \mathbf{z}^1 = [0 \ 0 \ -2] \cdot [1 \ 2 \ 1]^t < 0$

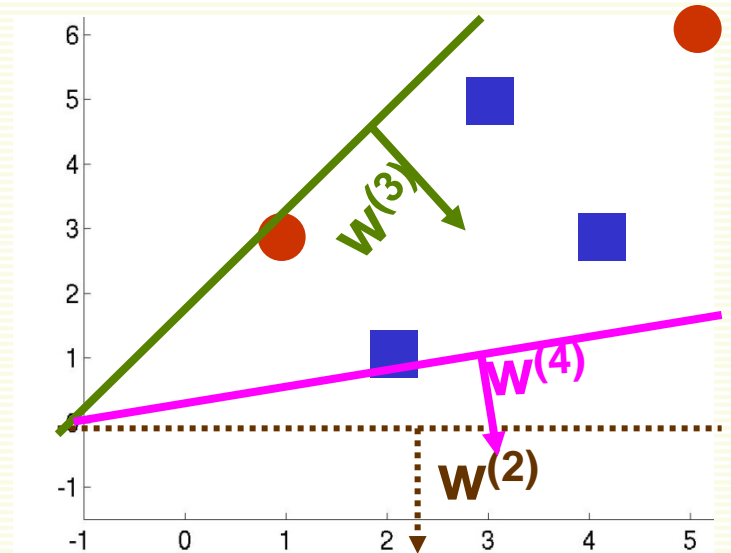
- *Update:* $\mathbf{a}^{(3)} = \mathbf{a}^{(2)} + \mathbf{z}_M = [0 \ 0 \ -2] + [1 \ 2 \ 1] = [1 \ 2 \ -1]$



Non-Linearly Separable Case

- $\mathbf{a}^{(3)} = [1 \ 2 \ -1]$
- rule is: $\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \mathbf{z}_M$

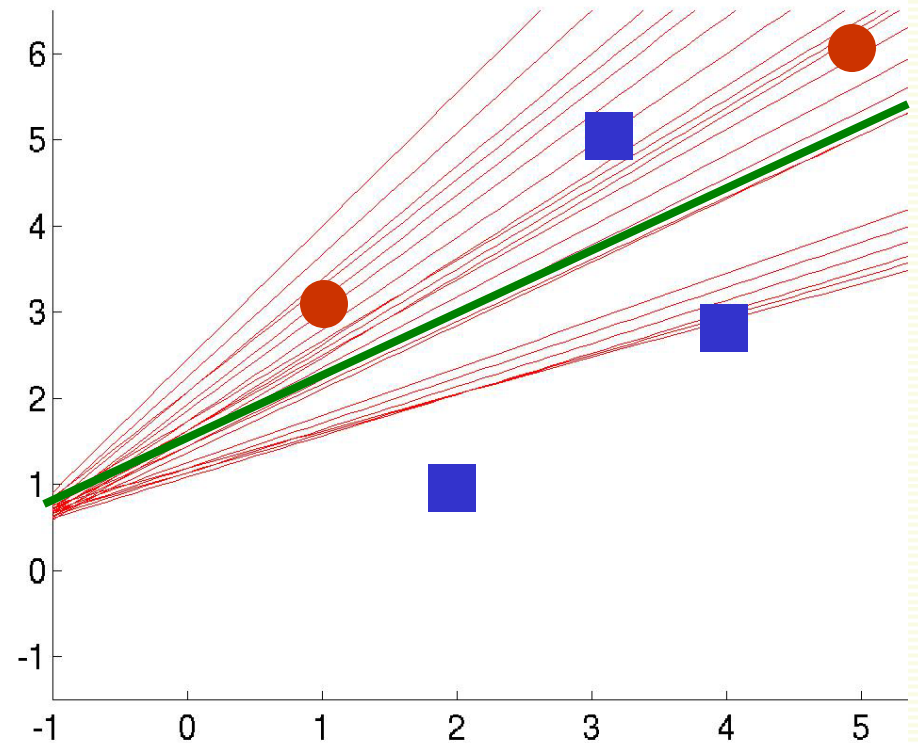
$$\mathbf{z}^1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \mathbf{z}^2 = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \quad \mathbf{z}^3 = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad \mathbf{z}^4 = \begin{bmatrix} -1 \\ -1 \\ -3 \end{bmatrix} \quad \mathbf{z}^5 = \begin{bmatrix} -1 \\ -5 \\ -6 \end{bmatrix}$$



- $\mathbf{a}^t \mathbf{z}^2 = [1 \ 4 \ 3] \cdot [1 \ 2 \ -1]^t = 6 > 0$
- $\mathbf{a}^t \mathbf{z}^3 = [1 \ 3 \ 5] \cdot [1 \ 2 \ -1]^t = 2 > 0$
- $\mathbf{a}^t \mathbf{z}^4 = [-1 \ -1 \ -3] \cdot [1 \ 2 \ -1]^t = 0$
- *Update:* $\mathbf{a}^{(4)} = \mathbf{a}^{(3)} + \mathbf{z}_M = [1 \ 2 \ -1] + [-1 \ -1 \ -3] = [0 \ 1 \ -4]$

Non-Linearly Separable Case

- We can continue this forever
 - there is no solution vector \mathbf{a} satisfying for all $\mathbf{a}^t \mathbf{z}_i > 0$ for all i
- Need to stop at a good point
- Solutions at iterations 900 through 915
- Some are good some are not
- How do we stop at a good solution?



Convergence of Perceptron Rules

1. Classes are linearly separable:
 - with fixed learning rate, both single sample and batch rules converge to a correct solution \mathbf{a}
 - can be any \mathbf{a} in the solution space
2. Classes are not linearly separable:
 - with fixed learning rate, both single sample and batch do not converge
 - can ensure convergence with appropriate variable learning rate
 - $\alpha \rightarrow 0$ as $\mathbf{k} \rightarrow \infty$
 - example, inverse linear: $\alpha = \mathbf{c}/\mathbf{k}$, where \mathbf{c} is any constant
 - also converges in the linearly separable case
 - no guarantee that we stop at a good point, but there are good reasons to choose inverse linear learning rate
 - Practical Issue: both single sample and batch algorithms converge faster if features are roughly on the same scale
 - see kNN lecture on feature normalization

Batch vs. Single Sample Rules

Batch

- True gradient descent, full gradient computed
- Smoother gradient because all samples are used
- Takes longer to converge

Single Sample

- Only partial gradient is computed
- Noisier gradient, therefore may concentrate more than necessary on any isolated training examples (those could be noise)
- Converges faster
- Easier to analyze

Minimum Squared Error Optimization

- Idea: convert to easier and better understood problem

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



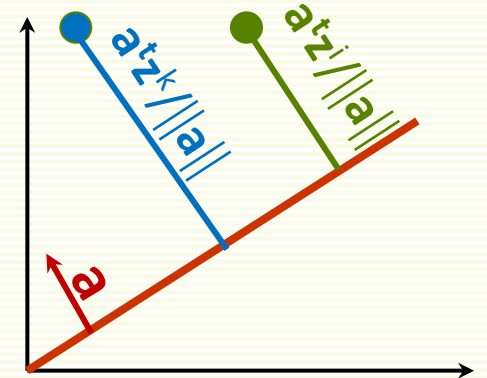
$\mathbf{a}^t \mathbf{z}^i = \mathbf{b}_i$ for all samples \mathbf{z}^i
solve system of linear equations

- MSE procedure
 - choose **positive** constants $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n$
 - try to find weight vector \mathbf{a} s.t. $\mathbf{a}^t \mathbf{z}^i = \mathbf{b}_i$ for all samples \mathbf{z}^i
 - if succeed, then \mathbf{a} is a solution because \mathbf{b}_i 's are positive
 - consider all the samples (not just the misclassified ones)

MSE: Margins

- By setting $\mathbf{a}^t \mathbf{z}^i = \mathbf{b}_i$, we expect \mathbf{z}^i to be at a relative distance \mathbf{b}_i from the separating hyperplane
- Thus $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n$ are expected relative distances of examples from the separating hyperplane
- Should make \mathbf{b}_i small if sample i is expected to be near separating hyperplane, and make \mathbf{b}_i larger otherwise
- In the absence of any such information, there are good reasons to set

$$\mathbf{b}_1 = \mathbf{b}_2 = \dots = \mathbf{b}_n = 1$$



MSE: Matrix Notation

- Solve system of n equations $\begin{cases} \mathbf{a}^t \mathbf{z}^1 = \mathbf{b}_1 \\ \vdots \\ \mathbf{a}^t \mathbf{z}^n = \mathbf{b}_n \end{cases}$
- Using matrix notation:

$$\begin{bmatrix} \mathbf{z}_0^1 & \mathbf{z}_1^1 & \cdots & \mathbf{z}_d^1 \\ \mathbf{z}_0^2 & \mathbf{z}_1^2 & \cdots & \mathbf{z}_d^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{z}_0^n & \mathbf{z}_1^n & \cdots & \mathbf{z}_d^n \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_d \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \end{bmatrix}$$

Z **a** **b**

- Solve a linear system $\mathbf{Za} = \mathbf{b}$

MSE: Exact Solution is Rare

- Solve a linear system $\mathbf{Za} = \mathbf{b}$
 - \mathbf{Z} is an n by $(d + 1)$ matrix
- Exact solution can be found only if \mathbf{Z} is nonsingular and square, in which case inverse \mathbf{Z}^{-1} exists
 - $\mathbf{a} = \mathbf{Z}^{-1} \mathbf{b}$
 - (number of samples) = (number of features + 1)
- if happens, guaranteed to find separating hyperplane
 - but almost never happens in practice

MSE: Approximate Solution

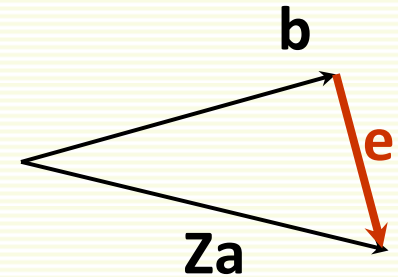
- Typically \mathbf{Z} is overdetermined
 - more rows (examples) than columns (features)

$$\boxed{\mathbf{z}} \boxed{\mathbf{a}} = \boxed{\mathbf{b}}$$

- No exact solution for $\mathbf{Za} = \mathbf{b}$ in this case
- Find an approximate solution \mathbf{a} , that is $\mathbf{Za} \approx \mathbf{b}$
 - approximate solution \mathbf{a} **does not** necessarily give a separating hyperplane in the separable case
 - but hyperplane corresponding to an approximate \mathbf{a} may still be a good solution

MSE Criterion Function

- MSE approach: find \mathbf{a} which minimizes the length of the error vector $\mathbf{e} = \mathbf{Za} - \mathbf{b}$



- Minimize the **minimum squared error** criterion function:

$$\mathbf{J}_s(\mathbf{a}) = \|\mathbf{Za} - \mathbf{b}\|^2 = \sum_{i=1}^n (\mathbf{a}^t \mathbf{z}^i - \mathbf{b}_i)^2$$

- Can be optimized exactly

MSE: Optimizing $J_s(\mathbf{a})$

$$J_s(\mathbf{a}) = \|\mathbf{Z}\mathbf{a} - \mathbf{b}\|^2 = \sum_{i=1}^n (\mathbf{a}^t \mathbf{z}^i - \mathbf{b}_i)^2$$

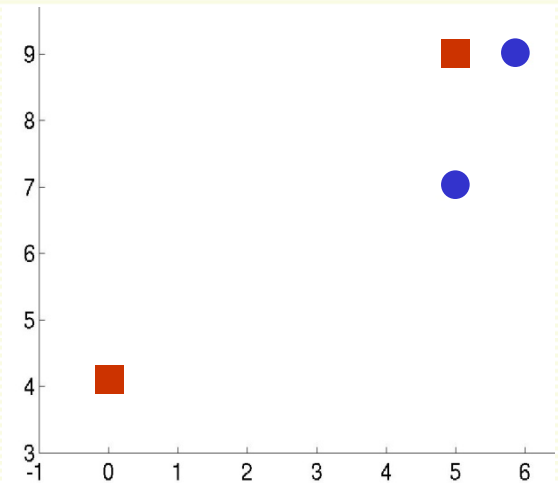
- Compute the gradient: $\nabla J_s(\mathbf{a}) = 2\mathbf{Z}^t(\mathbf{Z}\mathbf{a} - \mathbf{b})$
- Set it to zero: $2\mathbf{Z}^t(\mathbf{Z}\mathbf{a} - \mathbf{b}) = 0$
- If $\mathbf{Z}^t\mathbf{Z}$ is non-singular, its inverse exists and can find a unique solution for $\mathbf{a} = (\mathbf{Z}^t\mathbf{Z})^{-1} \mathbf{Z}^t\mathbf{b}$
- In Matlab
 - $\mathbf{a} = \mathbf{Z} \backslash \mathbf{b}$
 - Or use **pinv** command (pseudo-inverse)

MSE: Example

- Class 1: (6 9), (5 7)
- Class 2: (5 9), (0 4)
- Add extra feature and “normalize”

$$\mathbf{z}^1 = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix} \quad \mathbf{z}^2 = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} \quad \mathbf{z}^3 = \begin{bmatrix} -1 \\ -5 \\ -9 \end{bmatrix} \quad \mathbf{z}^4 = \begin{bmatrix} -1 \\ 0 \\ -4 \end{bmatrix}$$

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

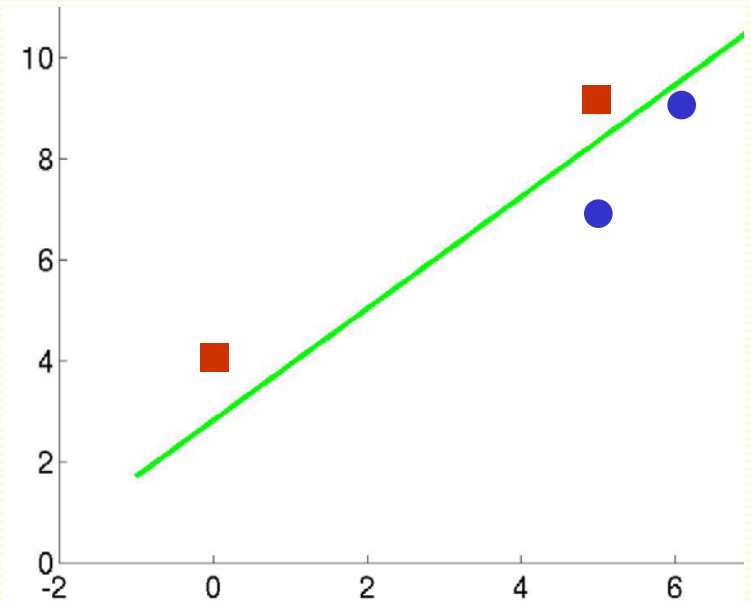


MSE: Example

- Choose $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
- Use $\mathbf{a} = \mathbf{Z} \backslash \mathbf{b}$ to solve in Matlab

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

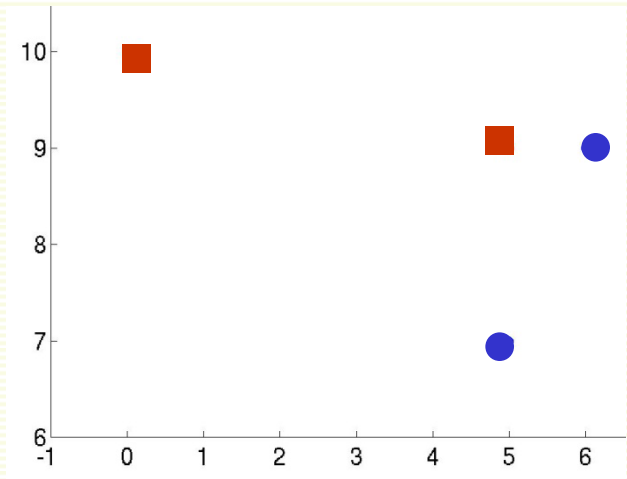
- Note \mathbf{a} is an approximation since $\mathbf{Z}\mathbf{a} = \begin{bmatrix} 0.4 \\ 1.3 \\ 0.6 \\ 1.1 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
- Gives a separating hyperplane since $\mathbf{Z}\mathbf{a} > 0$



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

MSE: Another Example

- Class 1: (6 9), (5 7)
- Class 2: (5 9), (0 10)
- One example is far compared to others from separating hyperplane

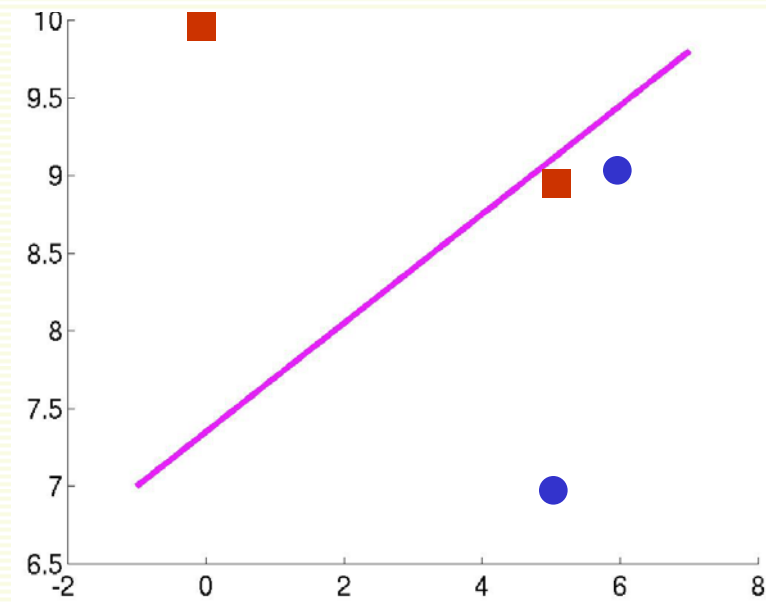


$$\mathbf{z}^1 = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix} \quad \mathbf{z}^2 = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} \quad \mathbf{z}^3 = \begin{bmatrix} -1 \\ -5 \\ -9 \end{bmatrix} \quad \mathbf{z}^4 = \begin{bmatrix} -1 \\ 0 \\ -10 \end{bmatrix}$$

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

MSE: Another Example Cont.

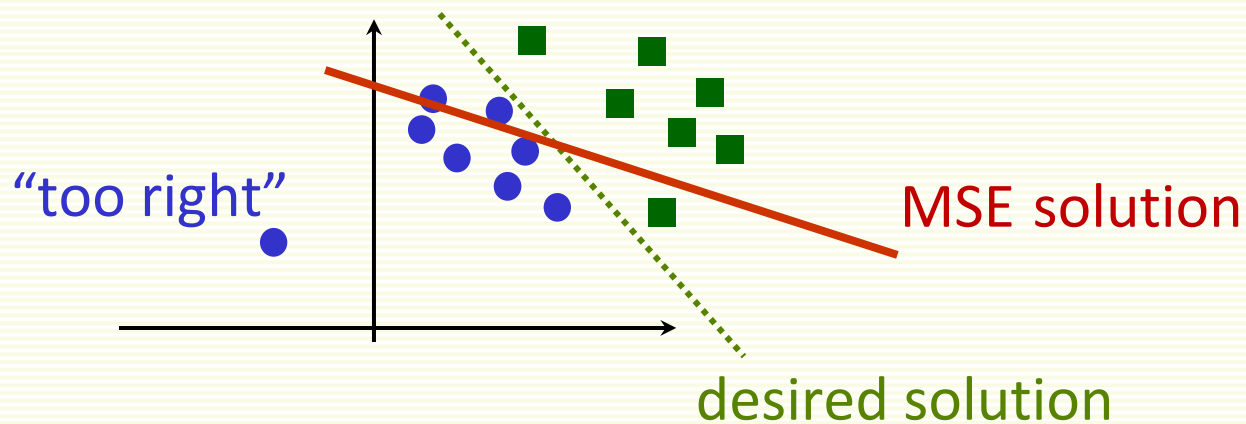
- Choose $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
- Solve $\mathbf{a} = \mathbf{Z} \backslash \mathbf{b} = \begin{bmatrix} 3.2 \\ 0.2 \\ -0.4 \end{bmatrix}$
- $\mathbf{Za} = \begin{bmatrix} 0.2 \\ 0.9 \\ -0.04 \\ 1.16 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$



- Does not give a separating hyperplane since $\mathbf{a}^t \mathbf{z}^3 < 0$

MSE: Problems

- MSE wants all examples to be at the same distance from the separating hyperplane
- Examples that are “too right”, i.e. too far from the boundary cause problems



- No problems with convergence though, both in separable and non-separable cases

MSE: Another Example Cont.

- If we know that 4th point is far from separating hyperplane

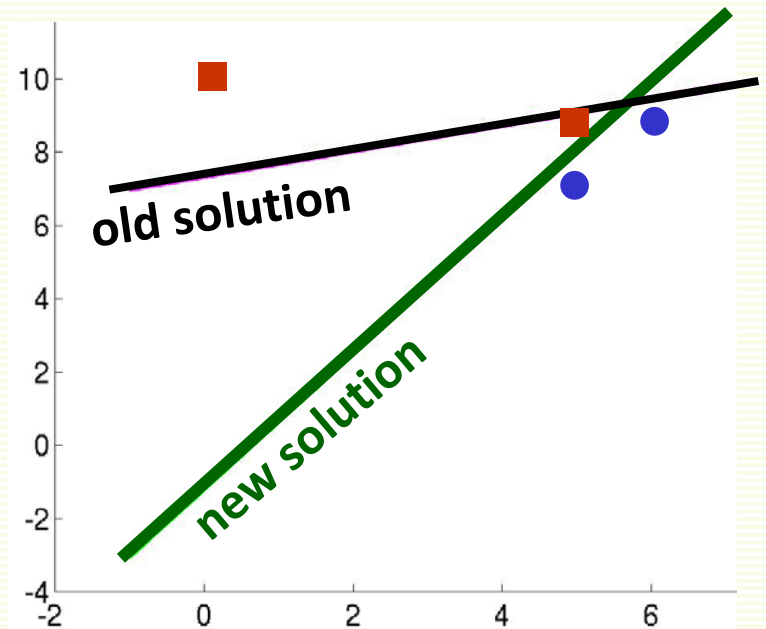
- in practice can look at points which are furthest from the decision boundary

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

- Set \mathbf{b}_i larger for such points: $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 10 \end{bmatrix}$

- Solve $\mathbf{a} = \mathbf{Z} \backslash \mathbf{b} = \begin{bmatrix} -1.1 \\ 1.7 \\ -0.9 \end{bmatrix}$

- $\mathbf{Za} = \begin{bmatrix} 0.9 \\ 1.0 \\ 0.8 \\ 10.0 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 10 \end{bmatrix} > 0$, therefore gives a separating hyperplane



More General Discriminant Functions

- Linear discriminant functions give simple decision boundary
 - try simpler models first
- 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
- Discriminant functions can be more general than linear
 - For example, polynomial discriminant functions
 - Decision boundaries more complex than linear
 - Later will look more at non-linear discriminant functions

Summary

- **Linear classifier** works well when examples are linearly separable, or almost separable
- **Two Training Approaches:**
 - **Perceptron Rules**
 - find a separating hyperplane in the linearly separable case
 - uses gradient descent for optimization
 - 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 Rules**
 - converges in separable and not separable case
 - can be optimized with pseudo-inverse
 - but may not find separating hyperplane even if classes are linearly separable