— LECTURE 5 —

PATTERN RECOGNITION

LINEAR DISCRIMINANT FUNCTIONS & Perceptron Learning

LINEAR DISCRIMINANT (LD)

$$g(x) = w^t x + w_0 \quad \leftarrow \text{LD}$$

where, $\vec{x} \in \mathbb{R}^d$ and $\vec{w} \in \mathbb{R}^d$; $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$, $\vec{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$, w_0 ↓ bias

$$g(x) = \langle \vec{w}, \vec{x} \rangle + w_0$$

- we can augment the bias term to the beginning of w:

$$w' = \begin{bmatrix} w_0 \\ \vec{w} \end{bmatrix}, \quad x' = \begin{bmatrix} 1 \\ \vec{x} \end{bmatrix} \Rightarrow \quad g(x) = \langle w', x' \rangle$$

[assume all $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$]

CLASSIFICATION:

Assume that we have two classes: w_1, w_2

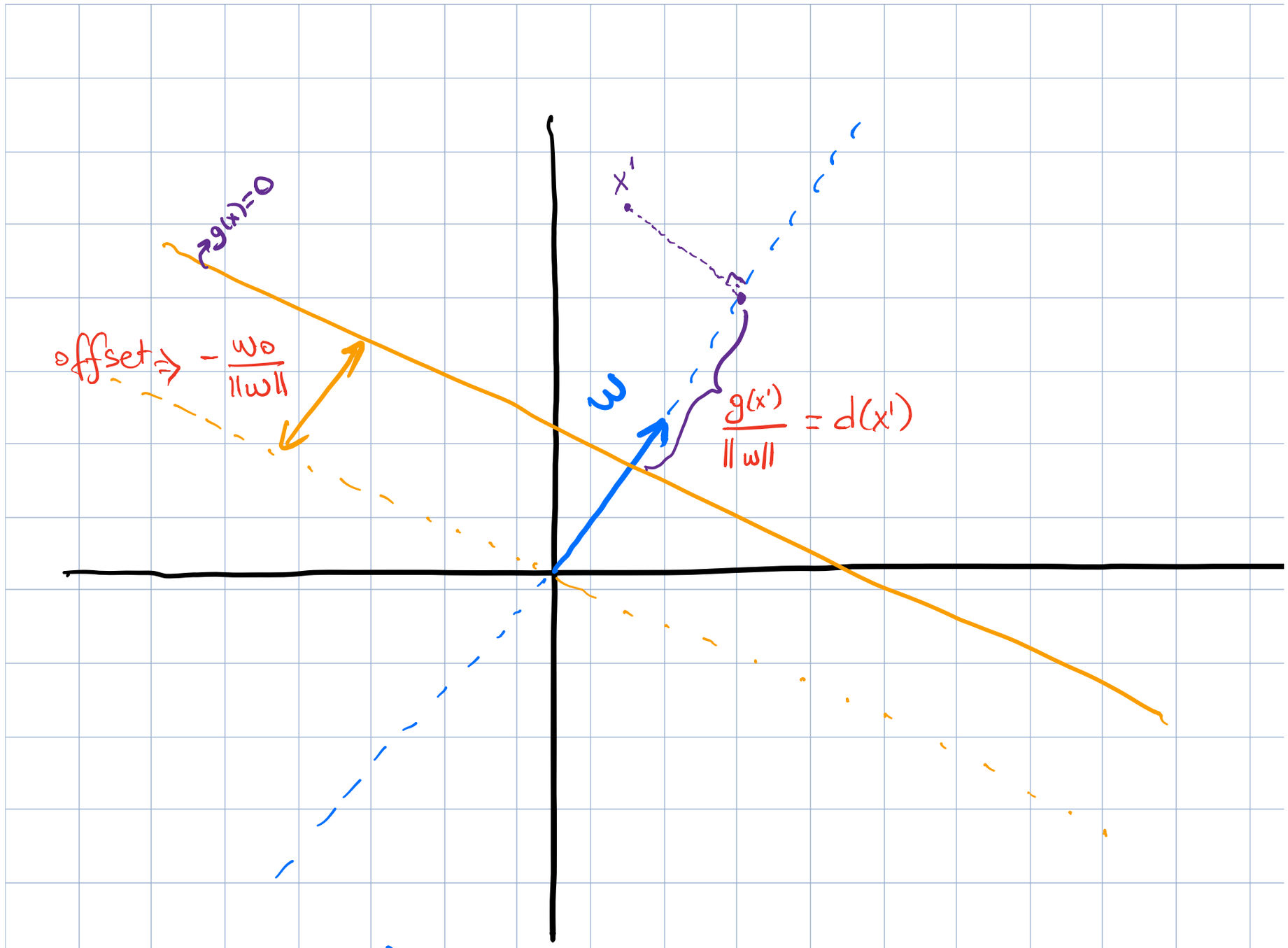
The decision rule of LD:

$$\begin{cases} w_1 \; ; \; \text{if} \quad g(x) > 0 \\ w_2 \; ; \; \text{if} \quad g(x) < 0 \end{cases}$$

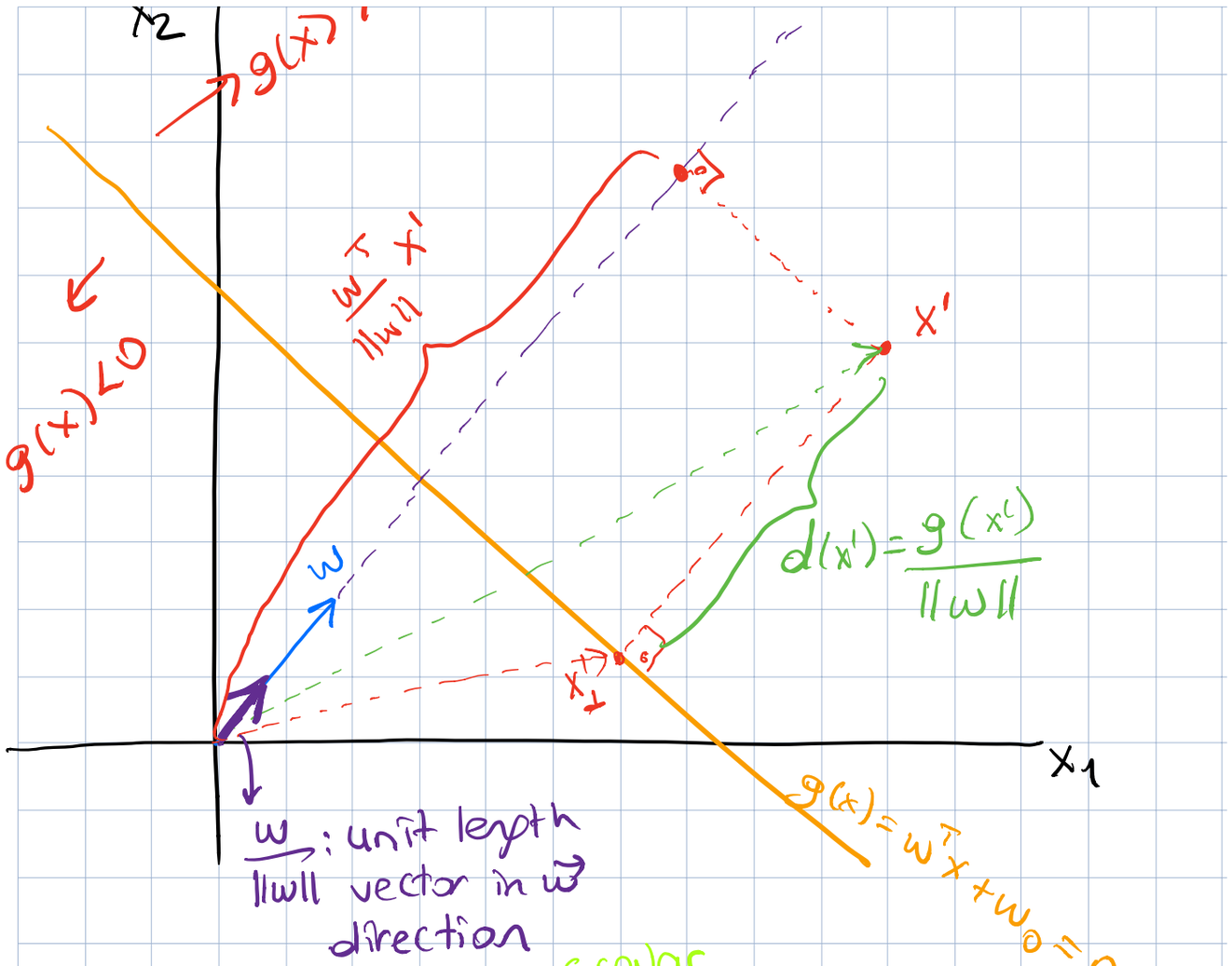
$g(x)=0 \rightarrow$ decision surface

↳ separates classes

- In linear case, $g(x)=0$ is a hyperplane
- $w \rightarrow$ the normal vector for $g(x)=0$ hp.



* $d(x') = \frac{g(x')}{\|w\|}$ → signed distance of x' to $g(x)=0$ hyperplane
* offset $= -\frac{w_0}{\|w\|}$ → signed distance of $g(x)=0$ hyperplane to the origin.



scalar

$$x' = x'_\perp + d(x') \cdot \frac{w}{\|w\|} \qquad \text{(Eqn 1)}$$

unit dir.

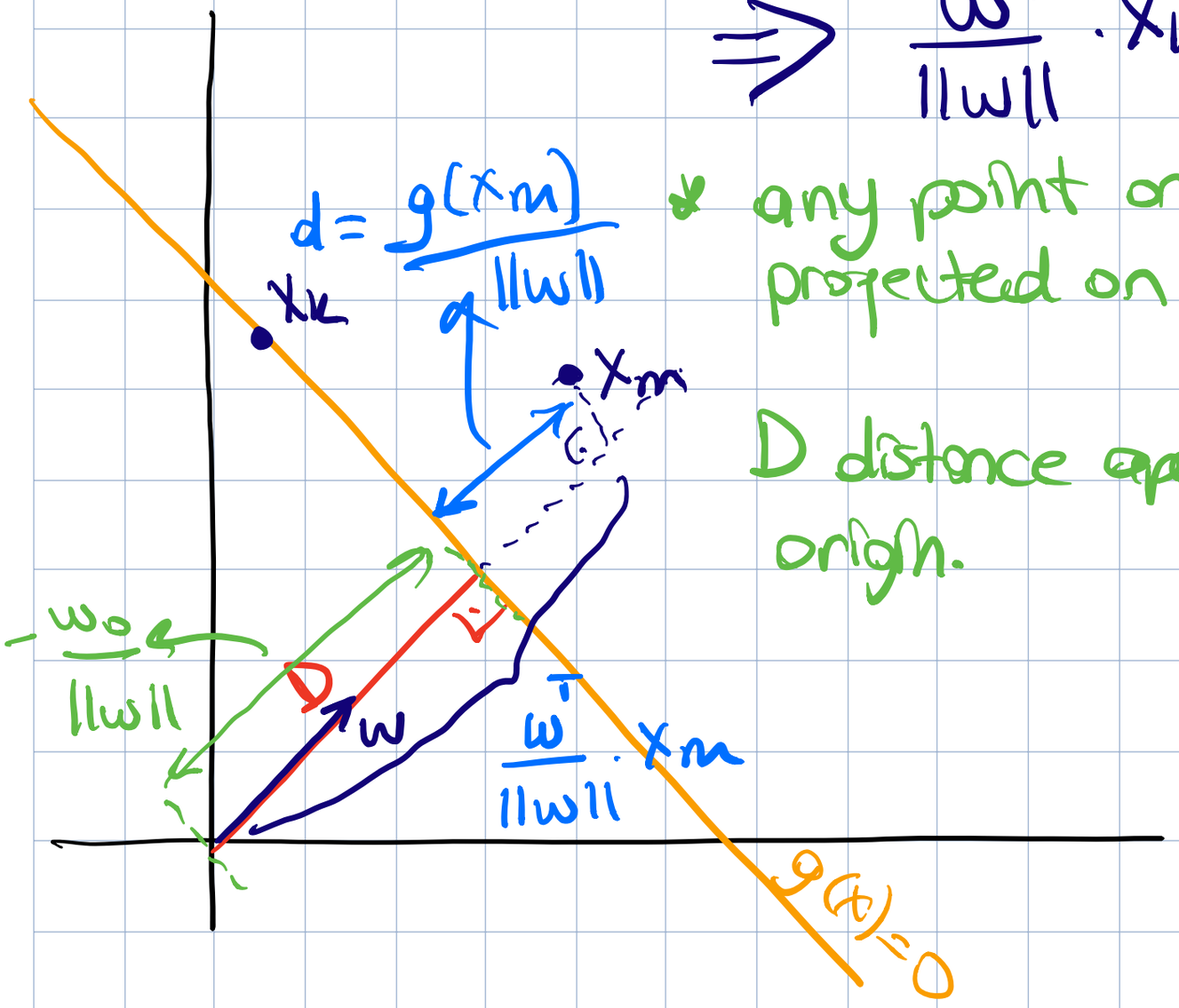
↳ multiply (Eqn 1) with w^T and add w_0:

$$\underbrace{w^T x' + w_0}_{g(x')} = w^T x'_\perp + w_0 + d(x')\|w\|$$

$$0 \Rightarrow g(x'_\perp) = 0$$

$$d(x') = \frac{g(x')}{\|w\|}$$

$$\Rightarrow \frac{w^T}{\|w\|} \cdot x_k = D$$



* any point on $g(x)=0$, when projected on $\frac{w}{\|w\|}$ will be D distance apart from the origin.

* $\frac{g(x_m)}{\|w\|} + D = \frac{w^T}{\|w\|} x_m$

$$d = \frac{w^T}{\|w\|} x_m - D$$

$\hookrightarrow -\frac{w_0}{\|w\|}$

Key Points:

- $g(x) = w^T x + w_0$: hyperplane

where w is the normal vector

- w_0 defines the position:

$$D = -\frac{w_0}{\|w\|}$$

- For any x':

- $g(x')$: projection of x' on the w dir.

- $d(x') = \frac{g(x')}{\|w\|}$ → signed dist. to $g(x)=0$ h.p.

HOW TO COMPUTE W AND W0?

How to Compute LD?

(1) Assuming a particular data distr. (i.e. Normal)

* Remember that we analyzed the discriminants based on a set of Normal distr. with different param. settings.

* Using the Bayes theorem, we constructed linear and quadratic discriminants.

For a C class problem, we defined

$g_i(x) = P(c_i|x)$; and classified a sample by selecting $\max g_i(x)$, where $i \in \{1,\dots,C\}$

$$\operatorname{argmax}_{c_i} (P(c_i|x)) = \operatorname{argmax}_{c_i} \frac{p(x|c_i)P(c_i)}{\sum_j p(x|c_j)P(c_j)}$$

$$= \operatorname{argmax}_{c_i} p(x|c_i)P(c_i)$$

$$= \operatorname{argmax}_{c_i} \left(\ln(p(x|c_i)) + \ln(P(c_i)) \right)$$

Recall:

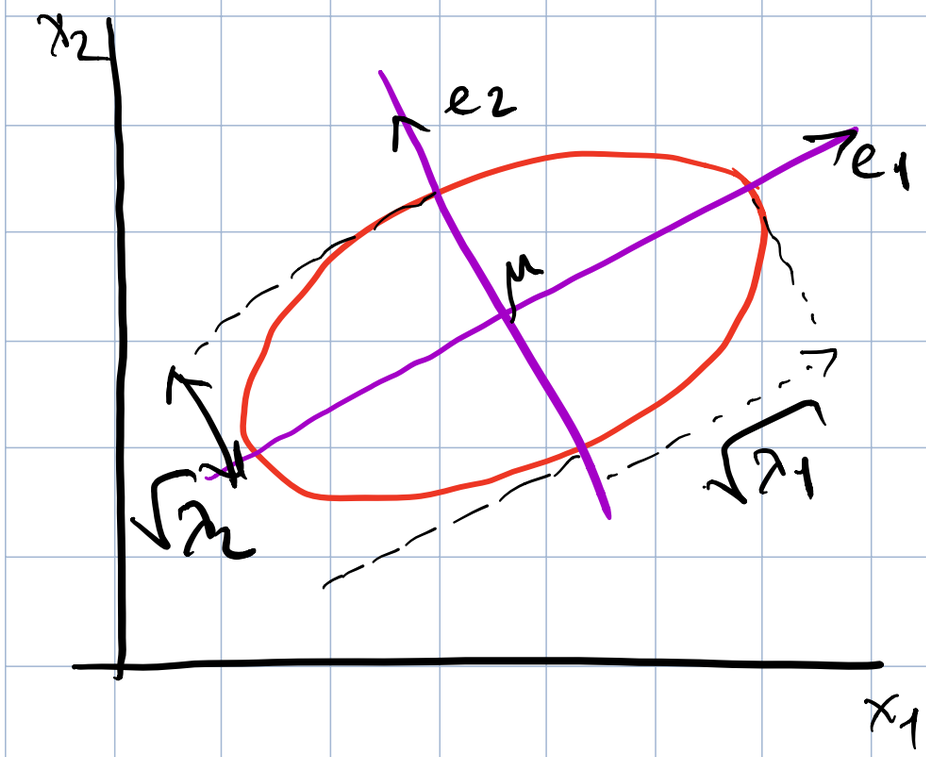
$$\text{argmax}_{c_i}\,(P(c_i|x)) = \text{argmax}_{c_i}(\ln(p(x|c_i)) + \ln(P(c_i)))$$

$$p(x|c_i) \sim N(\mu_i, \Sigma_i) = \overbrace{\frac{1}{(2\pi)^{d/2}\,|\Sigma_i|^{1/2}}}^{M}\; e^{-\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i)}$$

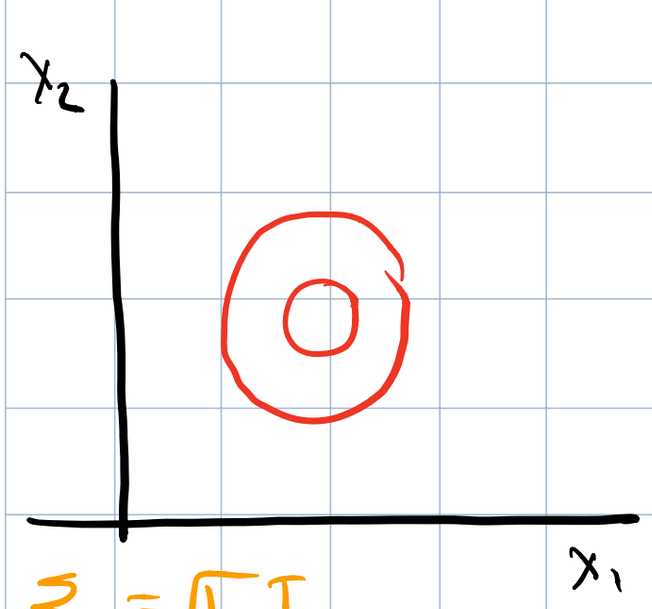
$$\text{argmax}_{c_i}(P(c_i|x)) = \text{argmax}_{c_i}(K) \quad ; \text{where:}$$

$$K = \ln(M) - \frac{1}{2}\underbrace{(x-\mu_i)^t \Sigma^{-1}(x-\mu_i)}_{\text{Mahalanobis distance from } \mu_i} + \ln(P(c_i))$$

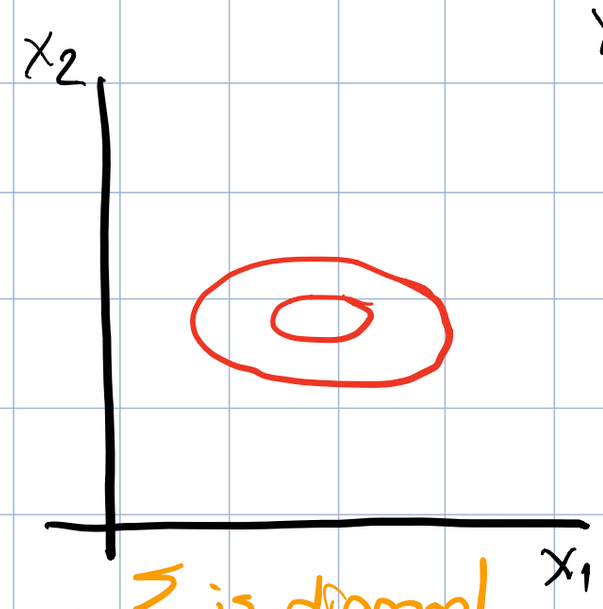
Recall: (Multivariate Normal Distr)



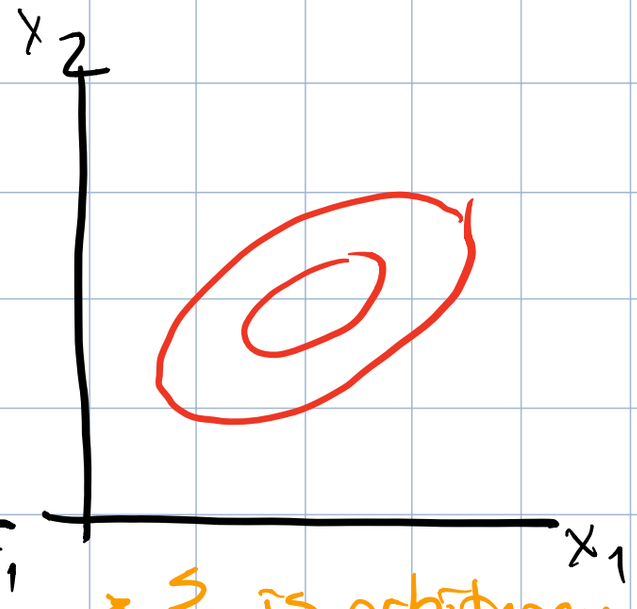
- The direction of the ellipsoid is determined by Σ
  - e_1 and e_2 are in the dir. of the eigenvectors of Σ
  - axes scales are prop. to eigenvalues of Σ: (λ_1, λ_2)



$\Sigma = \sigma I$



- Σ is diagonal
- $e_1 // x_1$, $e_2 // x_2$



- Σ is arbitrary (non-zero covariance)

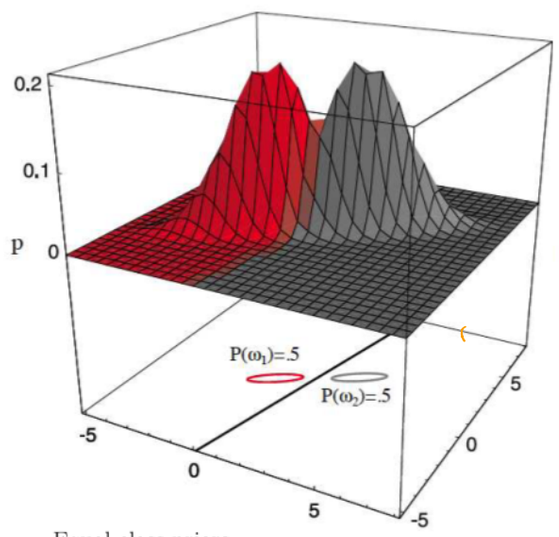
Depending on the shape of Σ, we get different discriminants:

(1) $\Sigma_i = \Sigma$:

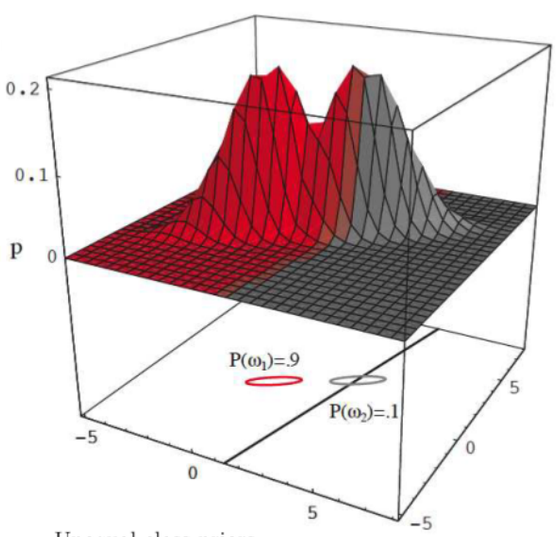
$$\text{argmax}(P(c_i|x)) = \text{argmax}_{c_i}\left(-\frac{1}{2}(x-\mu_i)^t \Sigma^{-1}(x-\mu_i) + \ln(P(c_i))\right)$$

* discriminant is linear since $x^t \Sigma^{-1} x$ is the same for all classes.
* simply classify the samples based on a simple function of Mahalanobis distance from the class centers and priors.

LD using distributional assumptions, $\Sigma_i = \Sigma$. Examples I



Equal class priors



Unequal class priors

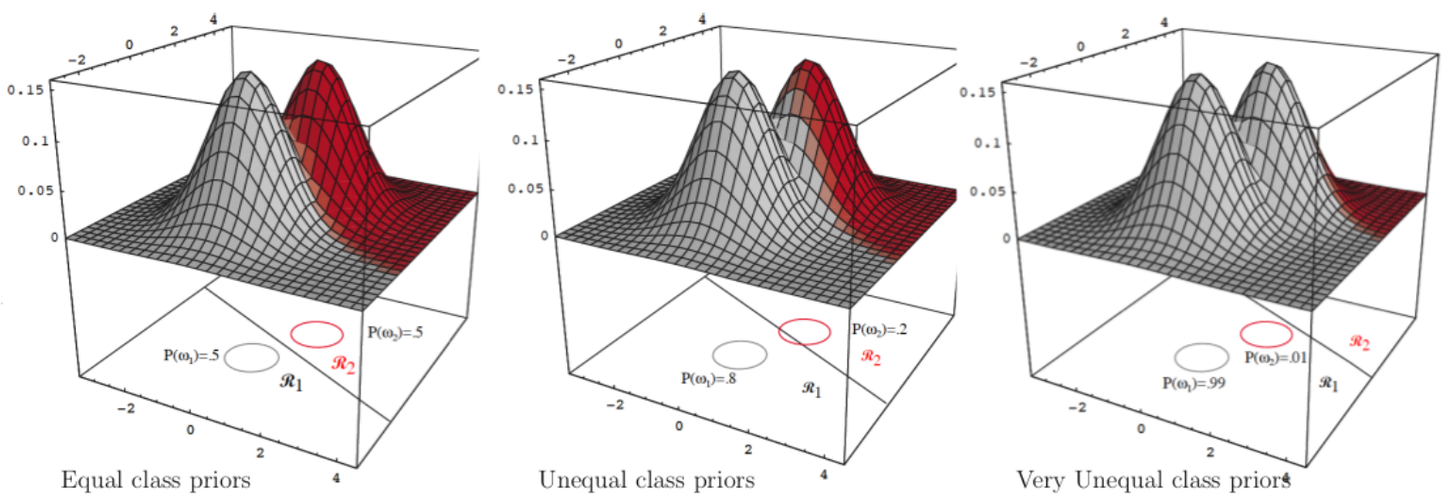
(2) $\Sigma_i = \sigma^2 I$

$$\text{argmax}_{c_i}\left(P(c_i|x)\right) = \text{argmax}_{c_i}\left(-\frac{\|x-\mu_i\|^2}{2\sigma^2} + \ln\left(P(c_i)\right)\right)$$

* we classify according to a function of Euclidean distance from the class centers and the priors

↳ a linear discriminant w.r.t. x.

LD using distributional assumptions, $\Sigma_i = \sigma^2 \boldsymbol{I}$



Equal class priors | Unequal class priors | Very Unequal class priors

How to Compute LD?

(2) Fisher's Linear Discriminant

* Project $x \in \mathbb{R}^d$ on to a single line for 2 classes.

* for c classes, project onto c-1 dim. space.

→ the direction for w is chosen such a way that the separation of the classes is maximum.

Define a cost function: $J(w)$

$$J(w) = \frac{\text{Between class scatter}}{\text{Within class scatter}} = \frac{S_B}{S_W}$$

Maximize $J(w)$: maximize S_B, minimize S_W

Fisher's LD (cont.)

$$S_B = |w^t \mu_1 - w^t \mu_2|$$

$$S_W = \sum_{x_i \in C_1} (w^t x_i - w^t \mu_1)^2 + \sum_{x_i \in C_2} (w^t x_i - w^t \mu_2)^2$$

spread of the projection for class C_1 : S_1

spread of the projection for class C_2 : S_2

$w^t \mu_i$ → projection of the mean μ_i of class c_i on the dir. of w.

We can rewrite $J(w)$: $\dfrac{w^t S_B w}{w^t S_W w}$

$$S_W = S_1 + S_2 \; ; \; S_i = \sum_{x \in c_i} (x - \mu_i)(x - \mu_i)^t$$

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^t$$

Fisher's LD (cont.)

$$J(w) = \frac{w^t S_B w}{w^t S_w w} \quad ; \text{where}$$

- $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^t$
- $S_w = \sum_{x \in C_1} (x - \mu_1)(x - \mu_1)^t + \sum_{x \in C_2} (x - \mu_2)(x - \mu_2)^t$

→ when we maximize $J(w)$, we get:

$$w = S_w^{-1} (\mu_1 - \mu_2)$$

Note that, Fisher's LD analysis do not compute w_0.
→ you can compute w_0 by assuming a Normal Distr. betw. the projected samples. (Ref: prev. slides)

How to compute LD ?

(3) PERCEPTRON

Perceptron learning algoritm can estimate w and w_0 of $g(x)$ if the classes are linearly separable.

The algorithm is simple:

if $g(x) > 0 \rightarrow c_1$
if $g(x) < 0 \rightarrow c_2$
} check the $sign(g(x))$

Hence, we can simplify the classification rule as:

if $sign(g(x)) = 1 \Rightarrow c_1$
if $sign(g(x)) = -1 \Rightarrow c_2$

PERCEPTRON (cont.)

* We want all x's to be classified correctly. Hence, define the cost function:

$$(w^t x + w_0)\, c > 0 \quad , \forall x$$

↳ class label for x

So;

- the cost is zero for correctly classified samples
- we'll try to minimize $-(w^t x + w_0)c$ for the misclassified samples

$$E(w, w_0) = -\sum_{x \in M} g(x)\, c$$

↳ misclassified sample set

PERCEPTRON (cont.)

The cost function:

$$E(\vec{w}, w_0) = -\sum_{x \in M} (\vec{w}^t \vec{x} + w_0) c$$

M: misclassified samples

Using gradient descent:

$$\vec{w}^{k+1} = \vec{w}^k - \alpha \nabla_w E(\vec{w}, w_0)$$

$$w_0^{k+1} = w_0^k - \alpha \nabla_{w_0} E(\vec{w}, w_0)$$

$$\nabla_w E(\vec{w}, w_0) = -c\vec{x}$$

$$\nabla_{w_0} E(\vec{w}, w_0) = -c$$

PERCEPTRON (cont.)

The Algorithm:

- Encode class labels for c_1 and c_2 as 1, -1

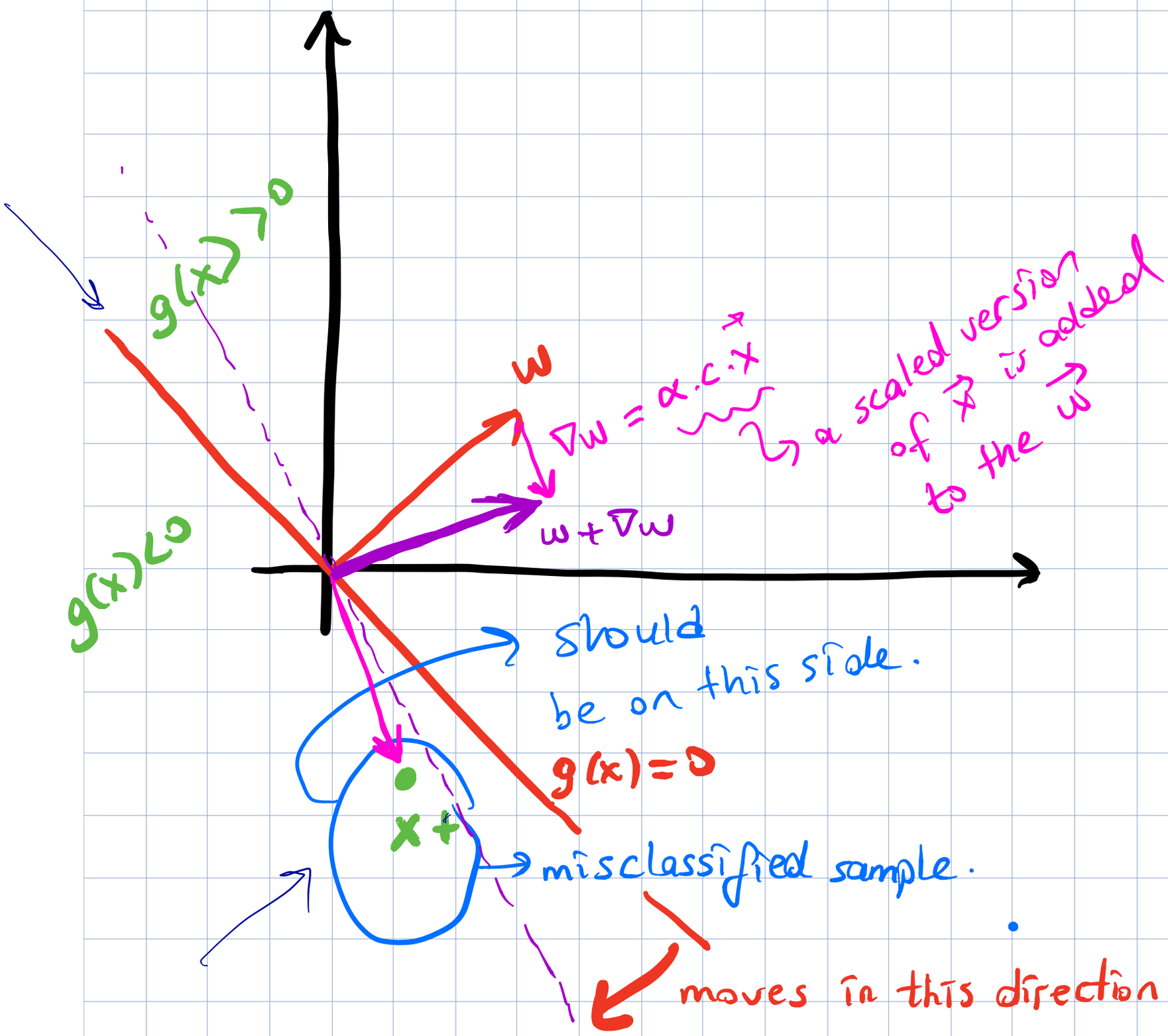
- Initialize $\vec{w}$ and w_0 randomly;
  α : learning rate to a small value

- For all x, repeat:

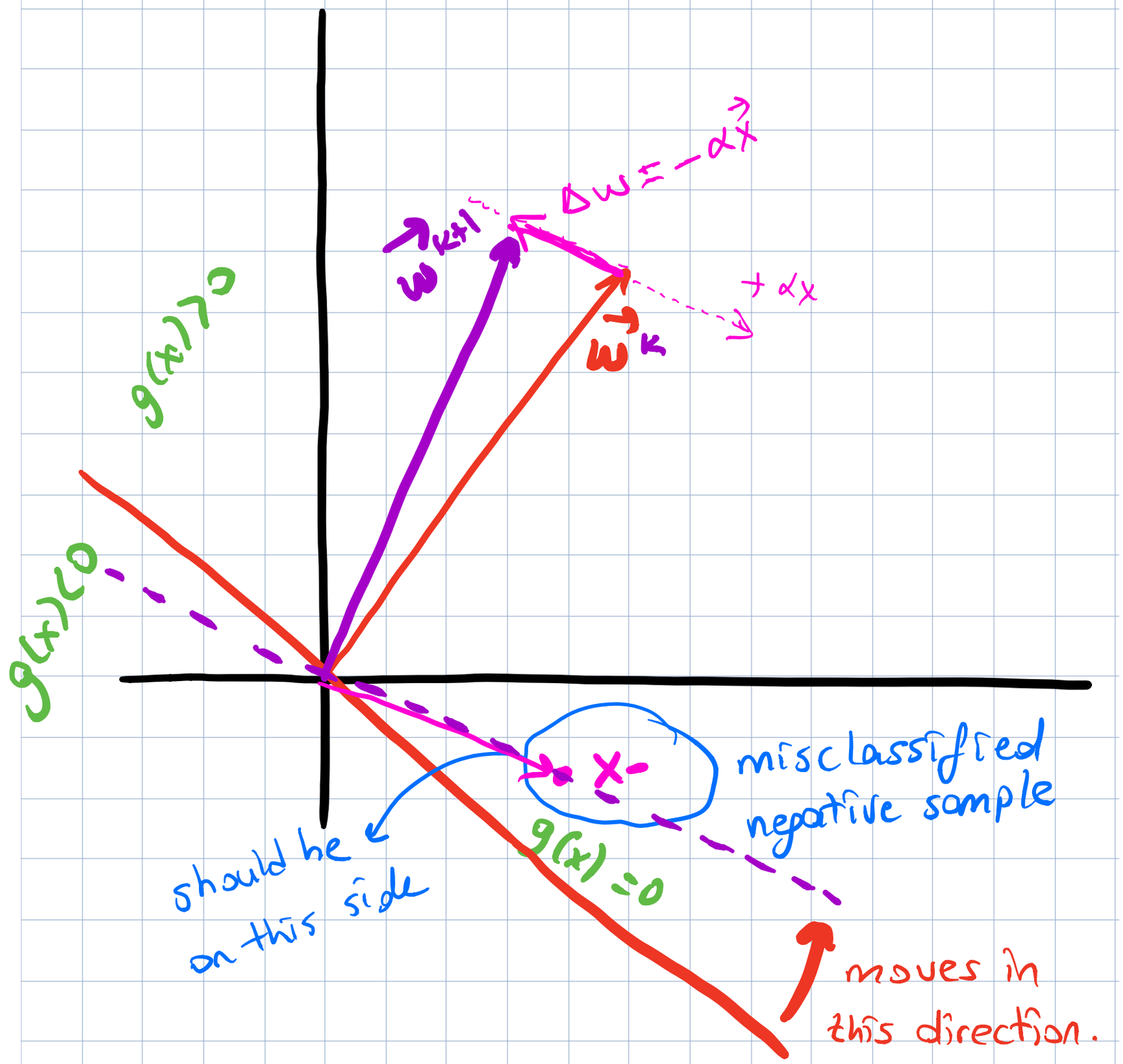
  score = sign $(g(x)) . c$
  if (score < 0) // update weights
  $w_i \leftarrow w_i + \alpha c x_i$
  $w_0 \leftarrow w_0 + \alpha c$

Until all x are classified correctly.

PERCEPTRON (cont.)



PERCEPTRON (cont.)



WINNOW

- A modified form of Perceptron Alg.

- Instead of updating the weights additively, update them multiplicatively

- Again, finds soln. only if the classes are linearly separable

The Algorithm:

- Encode the classes c_1, c_2 as $\{1, -1\}$
- Init $\vec{w}, w_0$ randomly;
  α: learning rate, $\beta > 1$

- For all x, repeat:

  score = sign $(g(x)) \cdot c$
  if (score < 0) // misclassified x
  $w_i \leftarrow w_i \times \Delta w_i$, $w_0 \leftarrow w_0 \times \Delta w_0$
  - $\Delta w_i = \beta^{\alpha c x_i}$
  - $\Delta w_0 = \beta^{\alpha c}$

Until all x are classified correctly.

BALANCED WINNOW

- Two weight vectors are utilized
- w^+, w^-: one for each of the two classes

The Algorithm:

- Initialize w^+, w^- randomly; $\beta > 1$
- for all x, repeat:

score $= \text{sign}(w^+ x + w_0^+ - (w^- x + w_0^-)) \cdot c$

if (score < 0) // misclassified x

$w_i^+ \leftarrow w_i^+ \times \Delta w_i^+$, $w_0^+ \in w_0^+ \times \Delta w_0^+$

- $\Delta w_i^+ = \beta^{c x_i}$, $\Delta w_0^+ = \beta^c$

$w_i^- \leftarrow w_i^- \times \Delta w_i^-$, $w_0^- \leftarrow w_0^- \times \Delta w_0^-$

- $\Delta w_i^- = \beta^{-c x_i}$, $\Delta w_0^- = \beta^{-c}$

Summary:

- Fisher's LD is problematic when $d > n$: S_w matrix is not invertible.
- In Fisher's LD, you need to compute w_0 seperately.
- Perceptron and winnow works for linearly seperable cases.
- Balanced winnow behaves well in high dim. data and works for linearly inseparable cases also.