



CS 6375

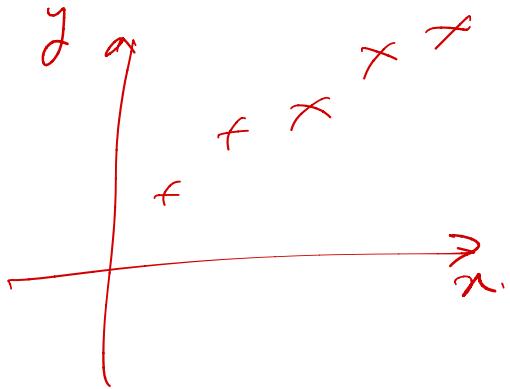
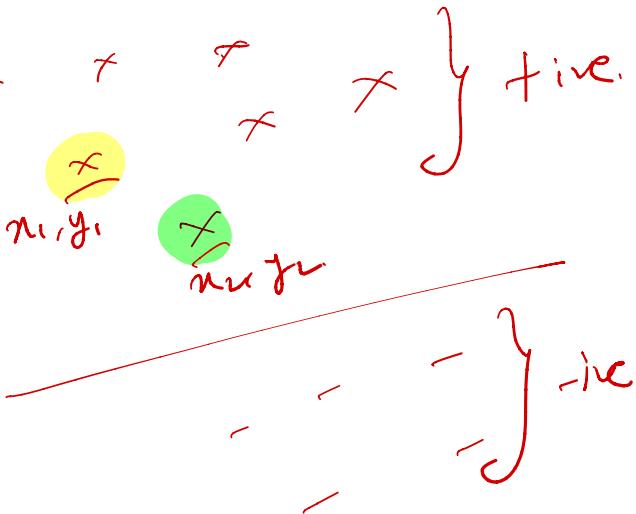
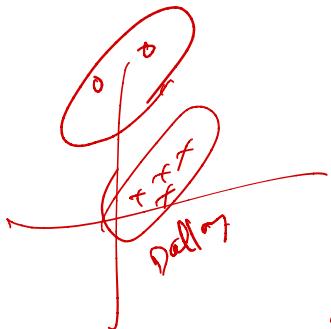
Binary Classification / Perceptron

Rishabh Iyer

University of Texas at Dallas

Part I: Recap of Supervised Learning, Linear Separation and Basics of Perceptron

"Independent & Identically Distributed"



class for

$$p(x_1, y_1, x_2, y_2) = p(x_1|y_1)p(x_2|y_2)$$

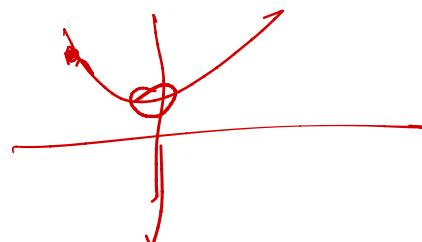
① Independent

② Identically Distributed

dist shift

History of Perceptron

- Formally introduced by Rosenblatt in 1958* ($1940^s, 50^s$)
- Introduced more like a *General-purpose Machine* rather than a classifier
 - This caused a heated controversy in the 1960's (NY times articles) etc.
- Soon, the limitations of perceptron's became evident
 - Works only in Linear separable cases
 - Cannot learn a simple XOR function
- However, these were the seeds for *Multi-Layer Perceptron's*, today known as Deep Neural Networks!

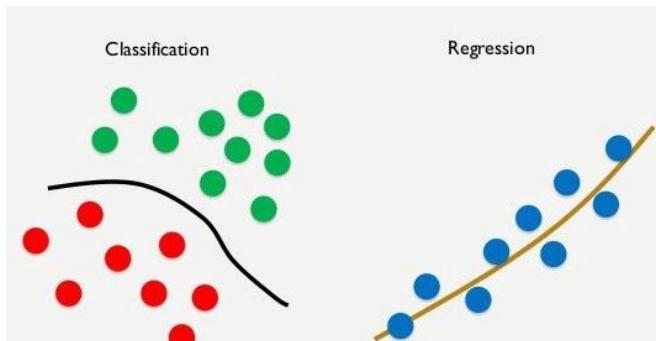


*Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408. doi:[10.1037/h0042519](https://doi.org/10.1037/h0042519)

Supervised Learning



- **Input:** $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$
 - $x^{(m)}$ is the m^{th} data item and $y^{(m)}$ is the m^{th} **label**
- **Goal:** find a function f such that $f(x^{(m)})$ is a “good approximation” to $y^{(m)}$
 - Can use it to predict y values for previously unseen x values



E.g. Regression

$$L(f(x), y) = [f(x) - y]^2$$

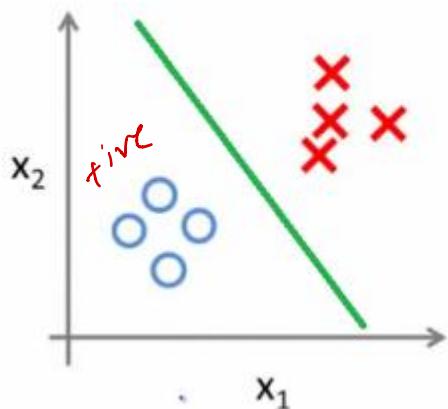
$$\sum_{i=1}^m L(f(x^{(i)}), y)$$

Supervised Learning

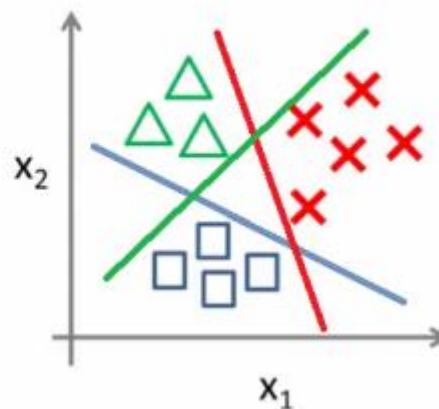
Classification vs Regression

- Input: pairs of points $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^d$
- Regression case: $y^{(m)} \in \mathbb{R}$ "lecture 2"
- Classification case: $y^{(m)} \in [0, k - 1]$ [k-class classification]
- If $k = 2$, we get Binary classification *(Today)*

Binary classification:



Multi-class classification:

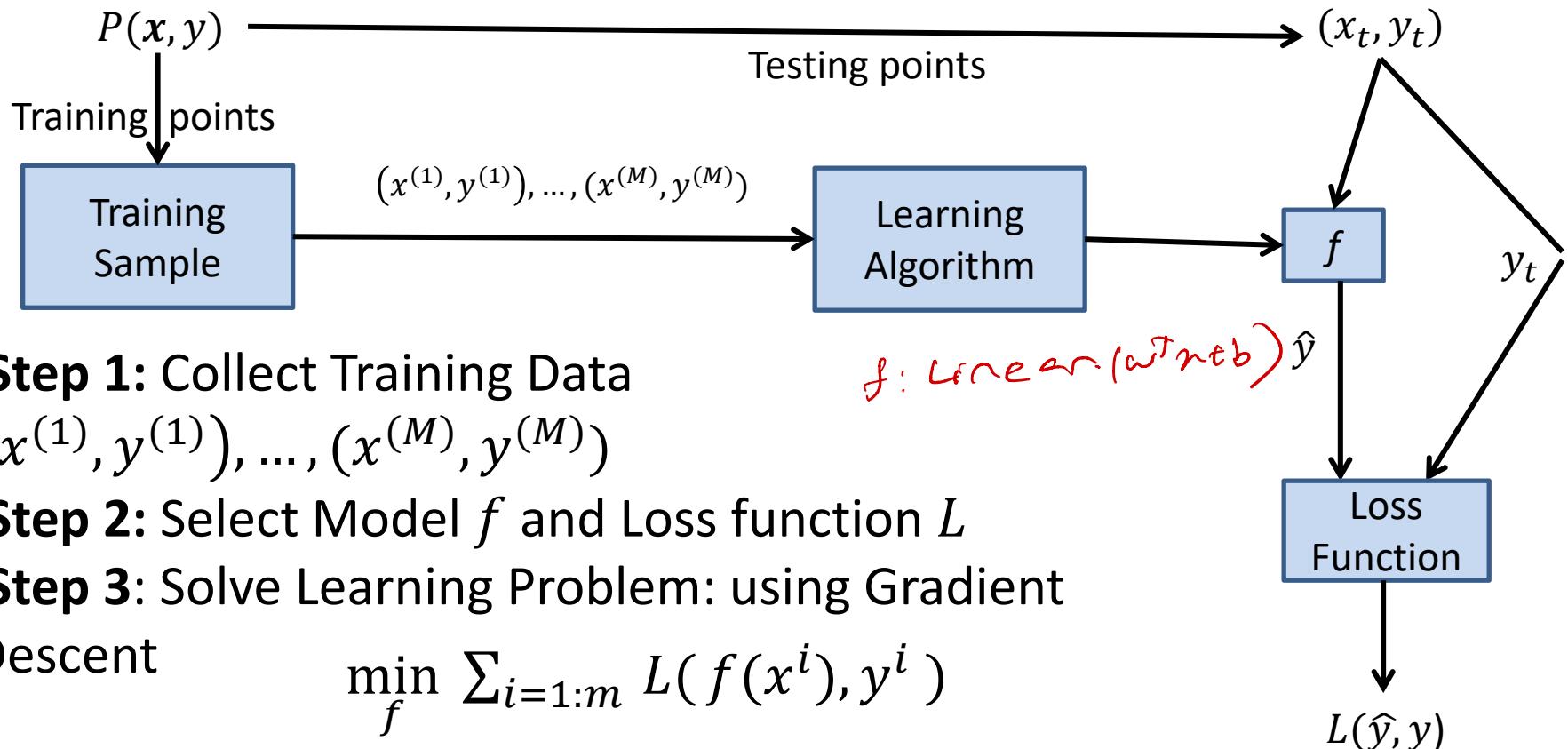


Recap: Hypothesis Space

- **Hypothesis space:** set of allowable functions $f: X \rightarrow Y$
- Goal: find the “best” element of the hypothesis space
- How do we measure the quality of f ?

$f: \underline{w^T n + b}$
parameters : (w, b)
 \uparrow \downarrow Int.
slope.

Recap: Supervised Learning Workflow



- **Step 1:** Collect Training Data $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$
- **Step 2:** Select Model f and Loss function L
- **Step 3:** Solve Learning Problem: using Gradient Descent

$$\min_f \sum_{i=1:m} L(f(x^i), y^i)$$
- **Step 4:** Obtain Predictions $\hat{y}_t = f(x_t)$ on all **Test Data**
- **Step 5:** Evaluation -- Measure the error $Err(\hat{y}_t, y_t)$ averaged over all **Test Data.**

$$\min_{w, b} \sum_{i=1}^m L_{\text{avg}}(f(x^{(i)}), y^{(i)})$$

Supervised Learning Workflow Cont...

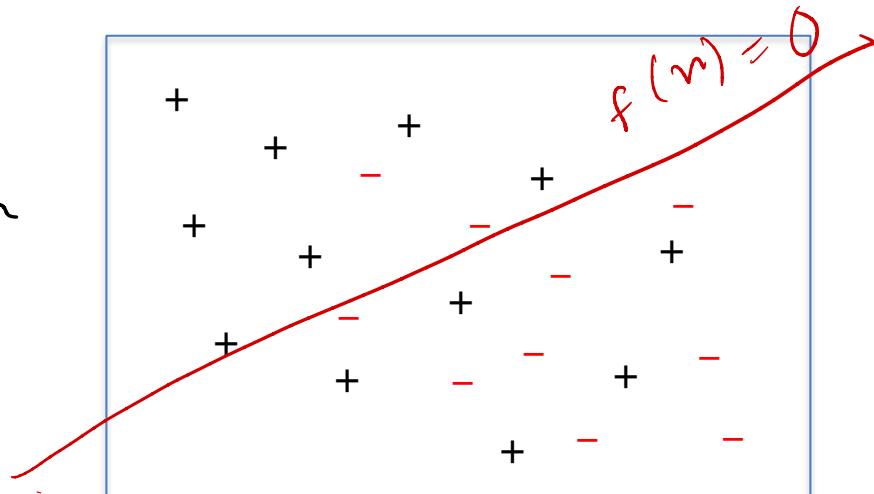


- Collect Training Data
- Select a hypothesis space (elements of the space are represented by a collection of parameters) (Select f)
- Choose a loss function (evaluates quality of the hypothesis as a function of its parameters) (Select L)
- Minimize loss function using gradient descent (minimization over the parameters) Optimization \rightarrow Gradient Descent
- Evaluate quality of the learned model using test data – that is, data on which the model was not trained (Eval / Test)

Binary Classification

- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \overbrace{\{-1, +1\}}^{\text{Labels}} \quad (k=2)$ $\underbrace{\text{features}}$
- We can think of the observations as points in \mathbb{R}^n with an associated sign (either +/- corresponding to 0/1)
- An example with $n = 2$

Eqⁿ of st. line
 $w^T x + b = 0$
Linear class fn



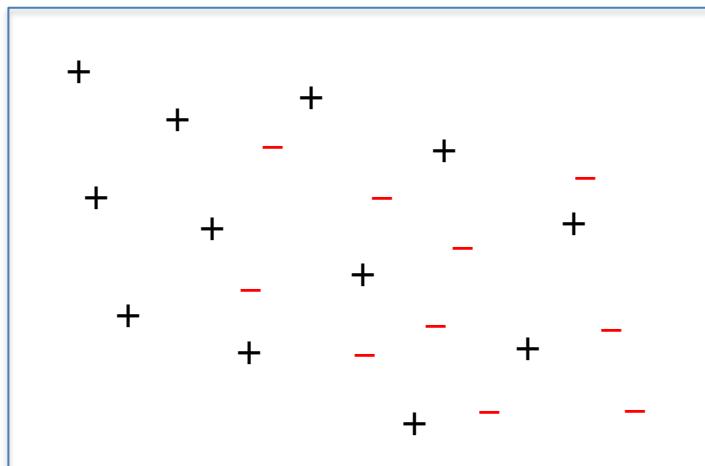
classifier

$f(w) \geq 0$
 \downarrow
 +ive.

$f(w) \leq 0$
 \downarrow
 -ive.

Binary Classification

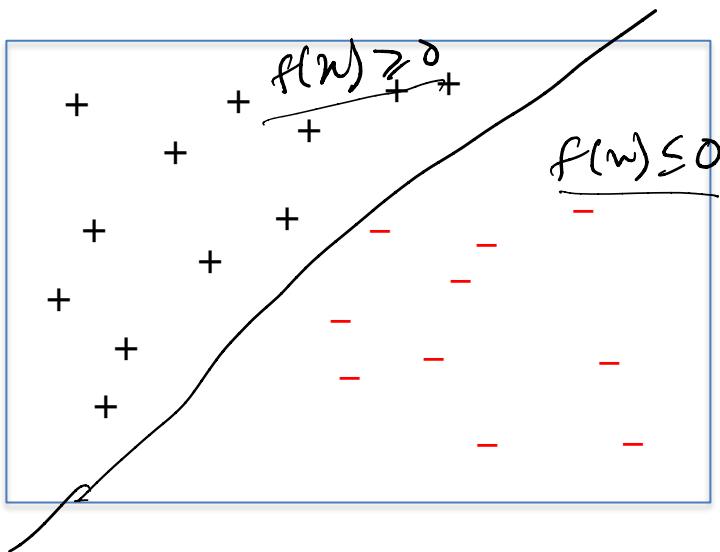
- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- We can think of the observations as points in \mathbb{R}^n with an associated sign (either +/- corresponding to 0/1)
- An example with $n = 2$



What is a good hypothesis space for this problem?

Binary Classification

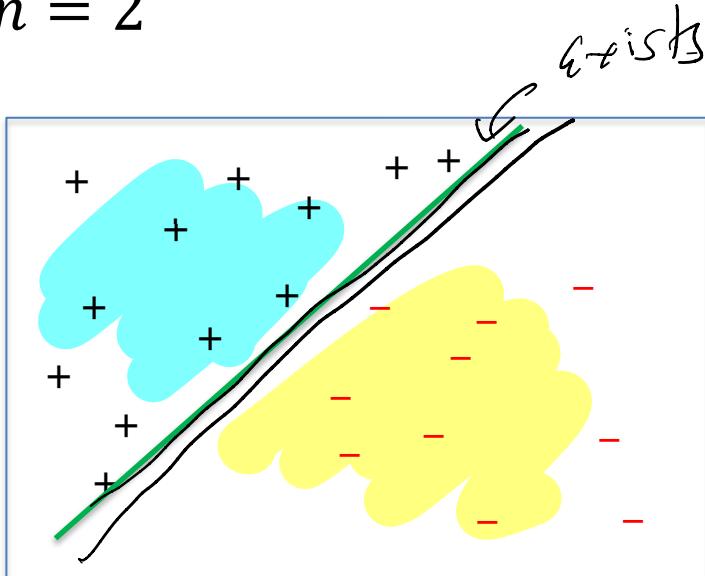
- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- We can think of the observations as points in \mathbb{R}^n with an associated sign (either +/- corresponding to 0/1)
- An example with $n = 2$



What is a good hypothesis space for this problem?

Binary Classification

- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- We can think of the observations as points in \mathbb{R}^n with an associated sign (either +/- corresponding to 0/1)
- An example with $n = 2$



In this case, we say
that the
observations are
linearly separable

Linear Separators



- In n dimensions, a hyperplane is a solution to the equation

$$\underline{w^T x + b = 0}$$

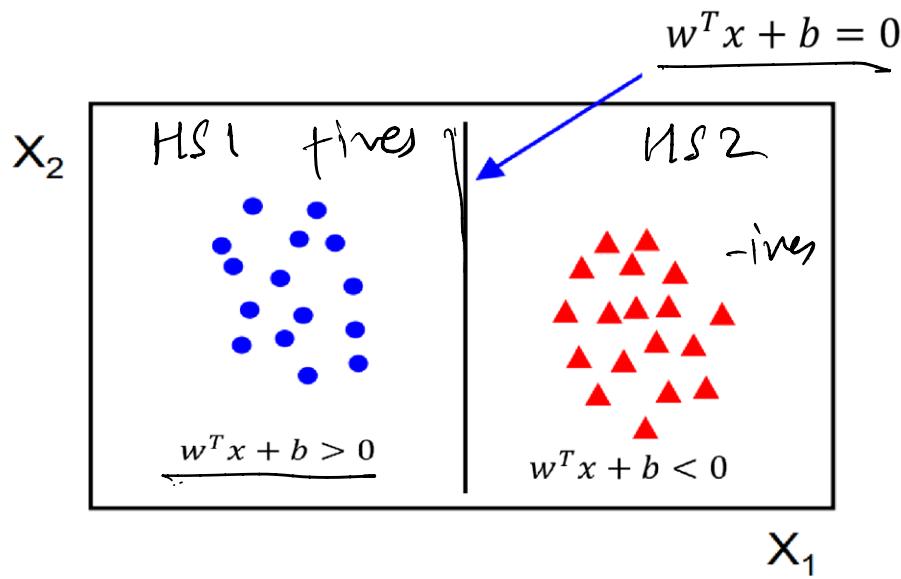
with $w \in \mathbb{R}^n, b \in \mathbb{R}$

- Hyperplanes divide \mathbb{R}^n into two distinct sets of points (called open halfspaces)

- Half Space 1: $w^T x + b > 0$
- Half Space 2: $w^T x + b < 0$

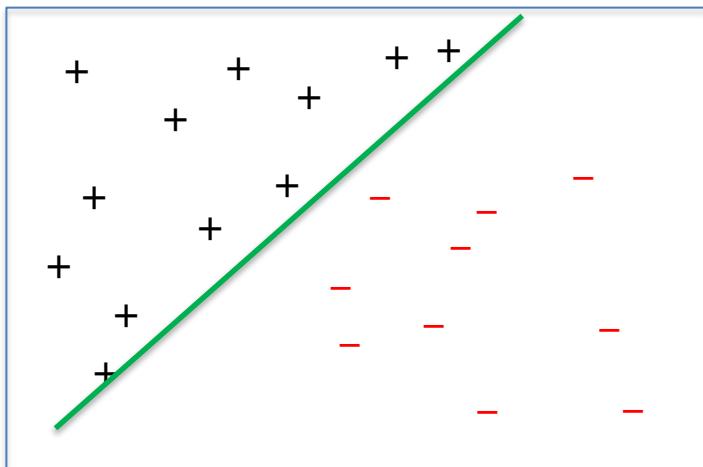
straight line-

- ① Hyperplane
② sep. Hyperplane



Binary Classification

- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- We can think of the observations as points in \mathbb{R}^n with an associated sign (either +/- corresponding to 0/1)
- An example with $n = 2$

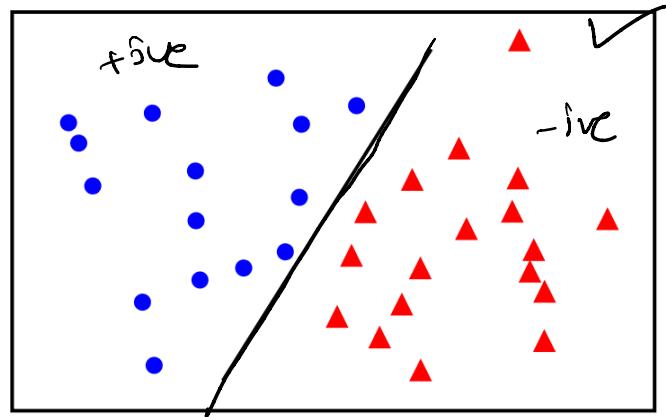
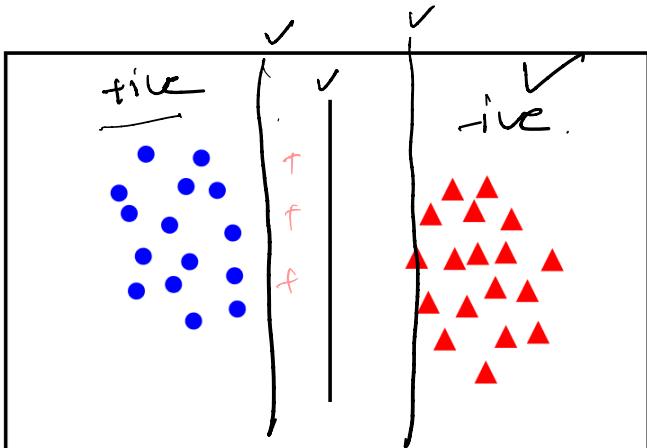


In this case, we say
that the
observations are
linearly separable

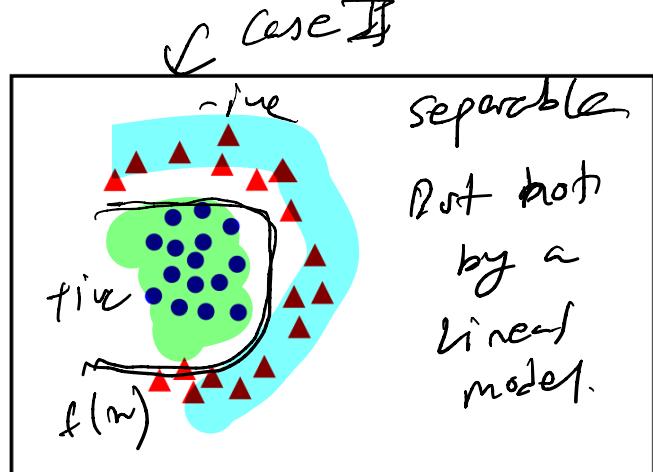
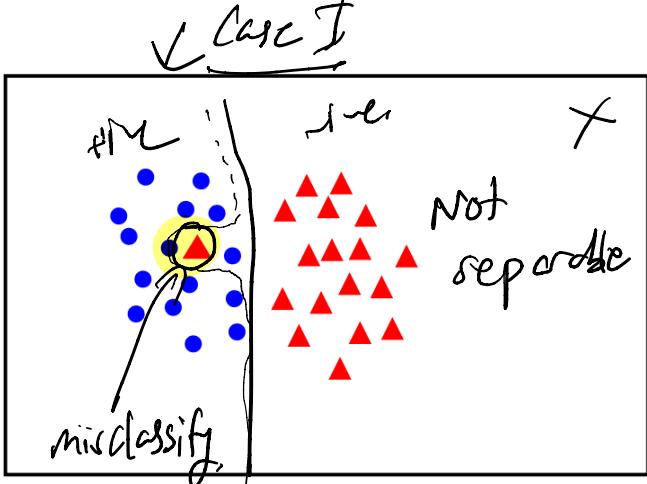
Linear Separable



linearly
separable



not
linearly
separable

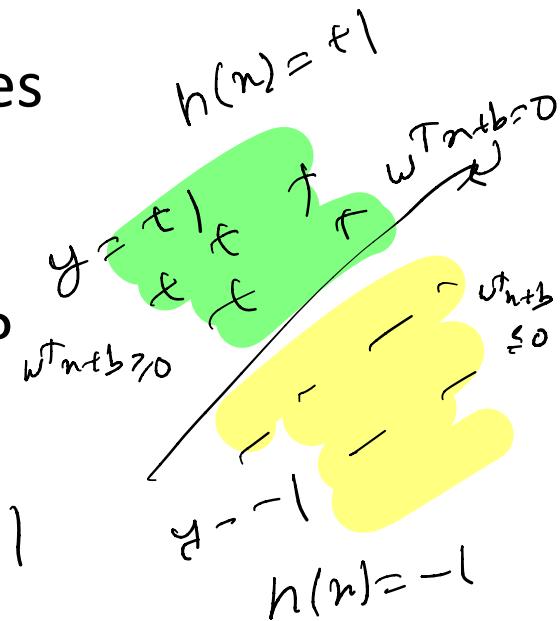


The Linearly Separable Case

- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- Hypothesis space: separating hyperplanes

$$h(n) = \underline{\text{f}(x)} = \underline{\text{sign}}(w^T x + b)$$

- How should we choose the loss function?



if $w^T n + b \geq 0$

$$h(n) = \text{sign}(w^T n + b) = +1$$

if $w^T n + b < 0$

$$h(n) = \text{sign}(w^T n + b) = -1$$

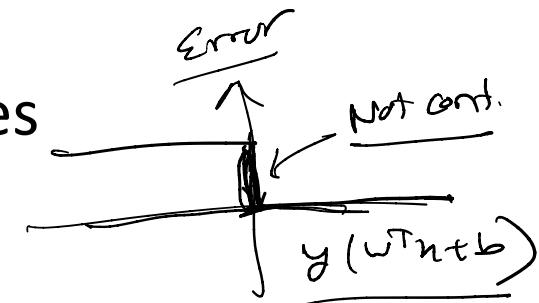
$$h(n) = \text{sign}(an^2 + bn + c) \leftarrow \text{quadratic}$$



The 0/1 Loss (Seperable Case)

- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- Hypothesis space: separating hyperplanes

$$\underbrace{f(x) = \text{sign}(w^T x + b)}$$



- How should we choose the loss function?

- Count the number of misclassifications ($\text{Error} = 1 - \text{Acc}$)

$$\text{zero/one loss} = \frac{1}{2} \sum_m |y^{(m)} - \text{sign}(w^T x^{(m)} + b)|$$

- Tough to optimize, gradient contains no information

The Perceptron Loss (Seperable Case)



- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$

- Hypothesis space: separating hyperplanes

$$f(x) = \text{sign}(w^T x + b)$$

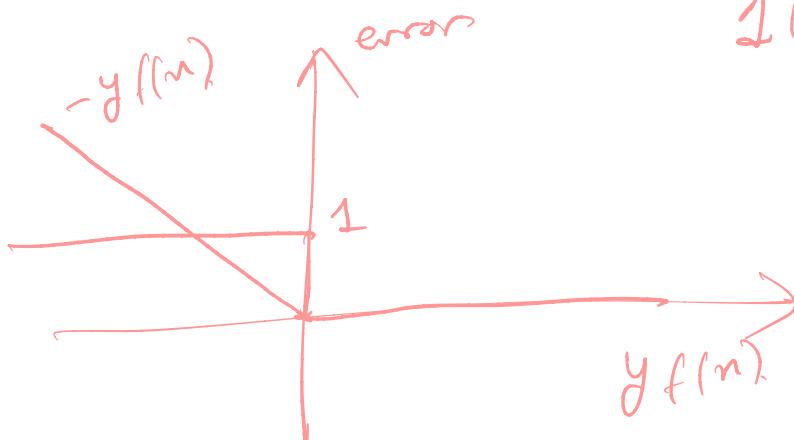
- How should we choose the loss function?

- Penalize misclassification linearly by the size of the violation

$$\text{perceptron loss} = \sum_m \max\{0, -y^{(m)}(w^T x^{(m)} + b)\}$$

- Modified hinge loss (this loss is convex, but not differentiable)

$$[y - (w^T x + b)]^2$$



$$\mathbb{1}(y f(m) < 0)$$

$$h(x) = \text{sign}(f(x))$$

$$w^T x + b$$

$$\uparrow \\ w^T x^{(m)} + b$$

$$-y f(m)$$

$$L(y, f(m)) = \max(0, -y f(m))$$

$$\mathcal{O}(l(y_1, f(m)) = \max(0, -y^{h(m)})$$

0/1 Loss Vs Perceptron Loss

- Zero/One Loss which counts the number of mis-classifications:

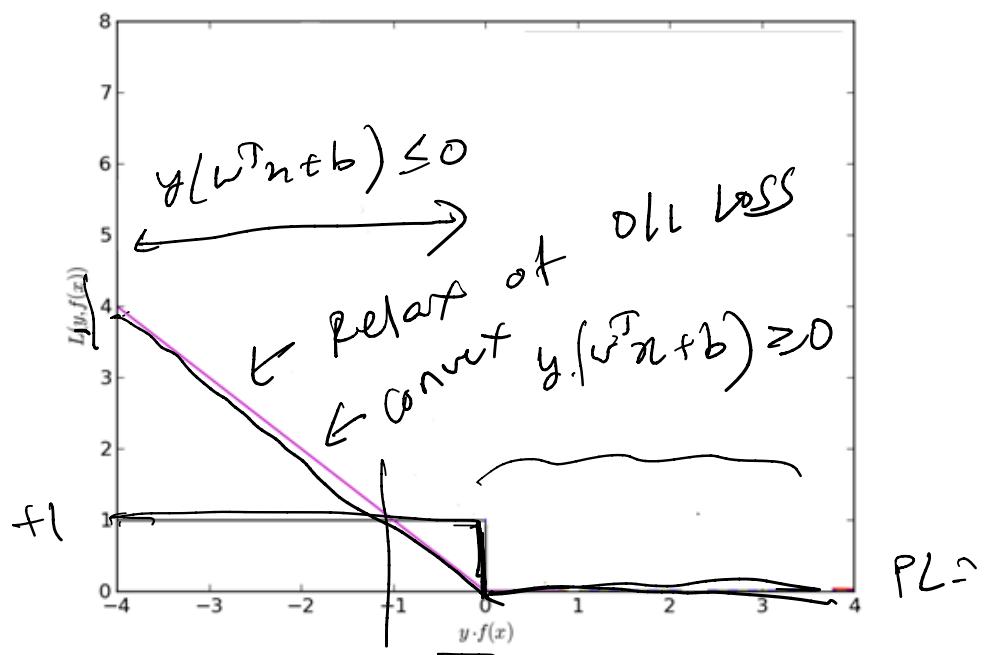
$$\text{zero/one loss} = \frac{1}{2} \sum_m |y^{(m)} - \text{sign}(w^T x^{(m)} + b)|$$

Not cont
 [Hard to compute gradients]

- Perceptron Loss:

$$\text{perceptron loss} = \sum_m \max\{0, -y^{(m)}(w^T x^{(m)} + b)\}$$

continuous
 [Not diff]



NP hard

The Perceptron Algorithm



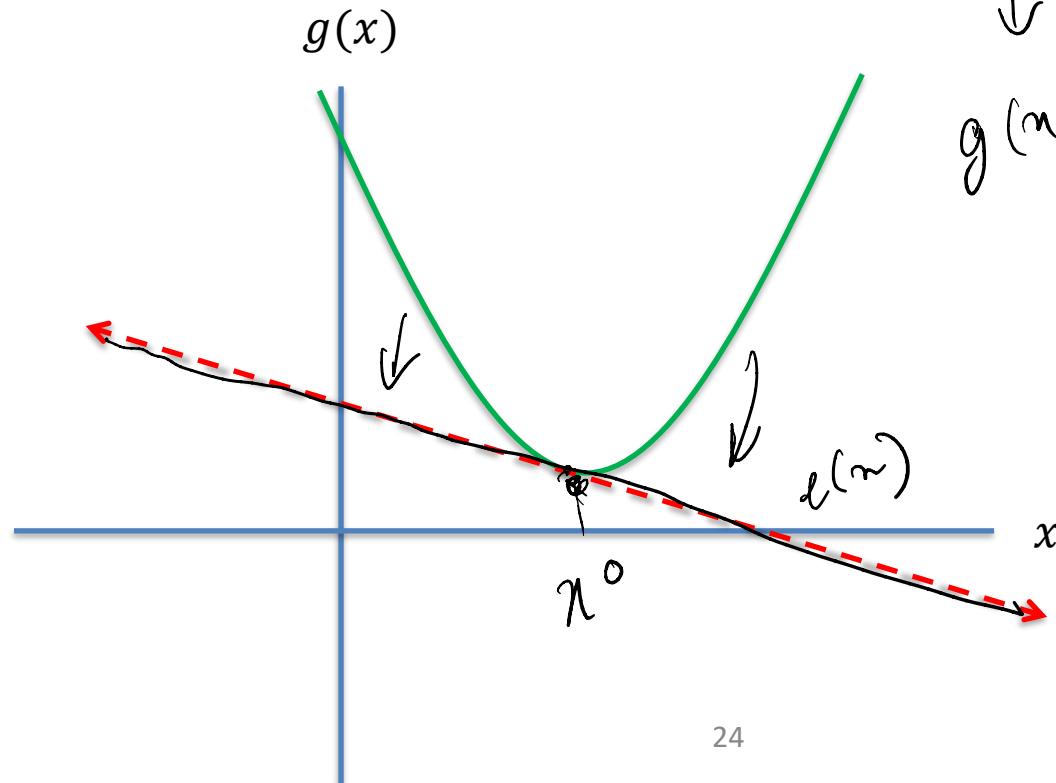
- Try to minimize the perceptron loss using gradient descent
 - The perceptron loss isn't differentiable, how can we apply gradient descent?
 - Need a generalization of what it means to be the gradient of a **convex** function

Part II: (Sub) Gradient Descent and Perceptron

Gradients of Convex Functions



- For a differentiable convex function $g(x)$ its gradients are **linear underestimators**

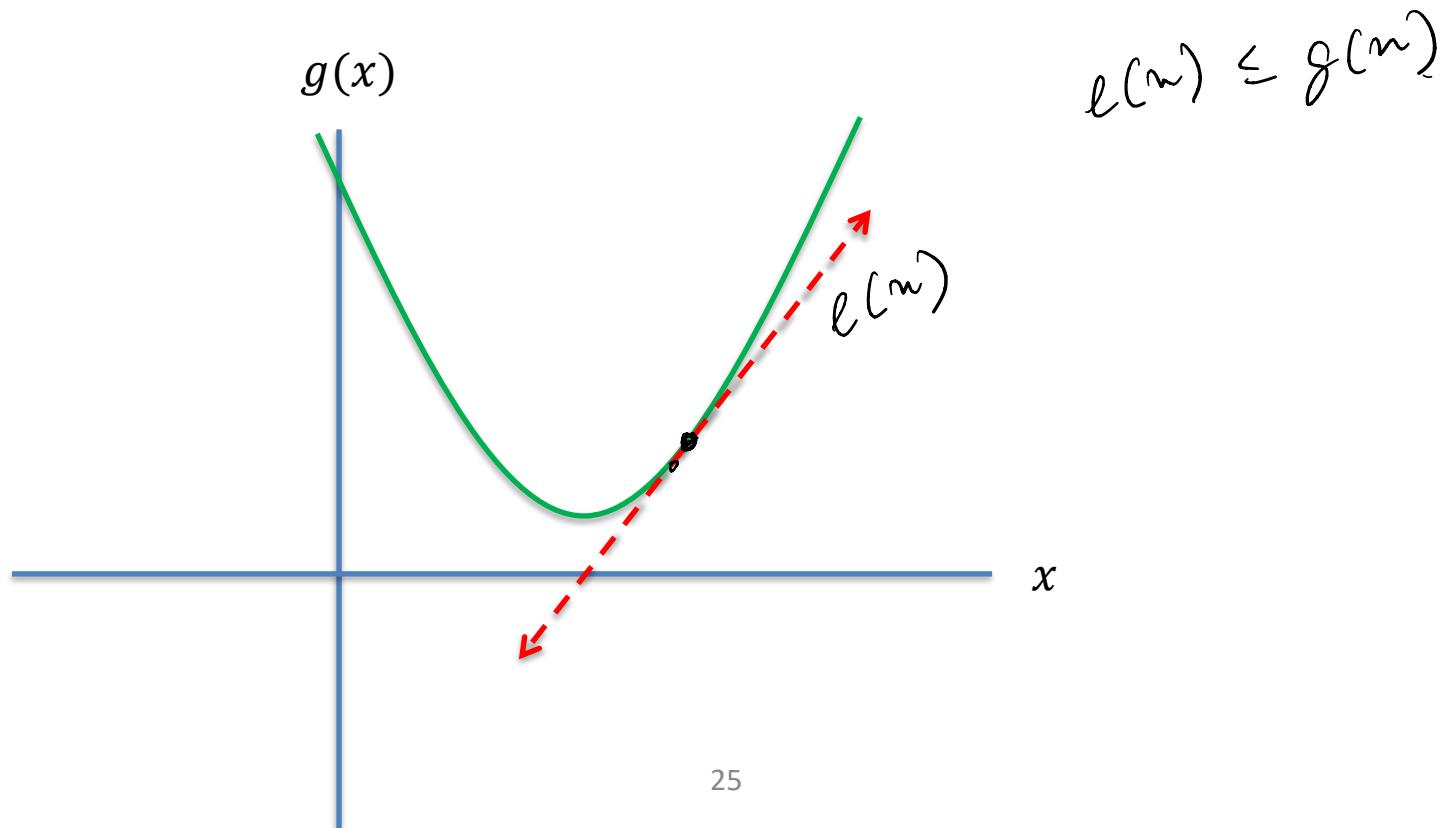


$$\begin{aligned} \ell(n) &\leq g(n) \\ \downarrow & \\ g(n_0) + \langle \nabla g(n_0), n - n_0 \rangle &\leq g(n) \end{aligned}$$

Gradients of Convex Functions



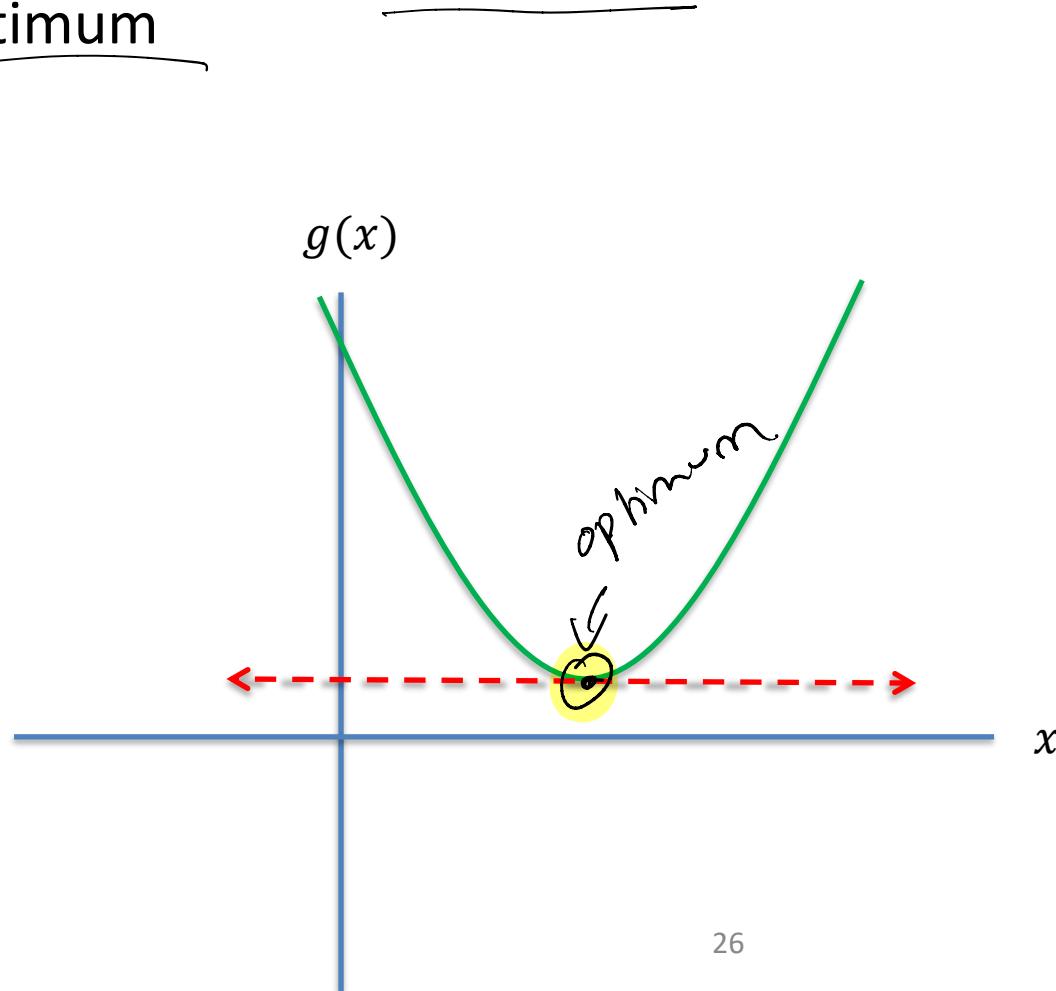
- For a differentiable convex function $g(x)$ its gradients are **linear underestimators**



Gradients of Convex Functions

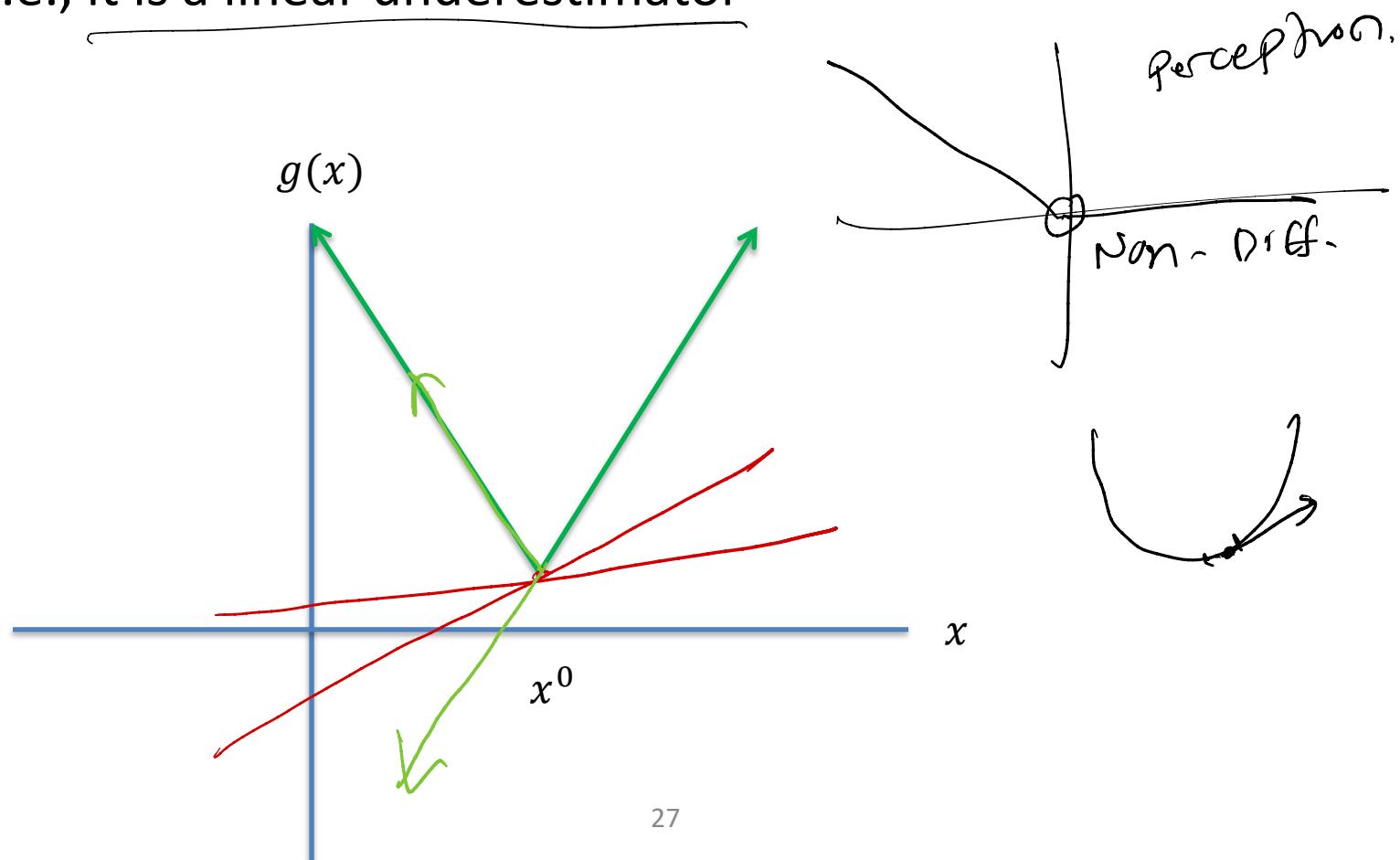


- For a differentiable convex function $g(x)$ its gradients are **linear underestimators**: zero gradient corresponds to a global optimum



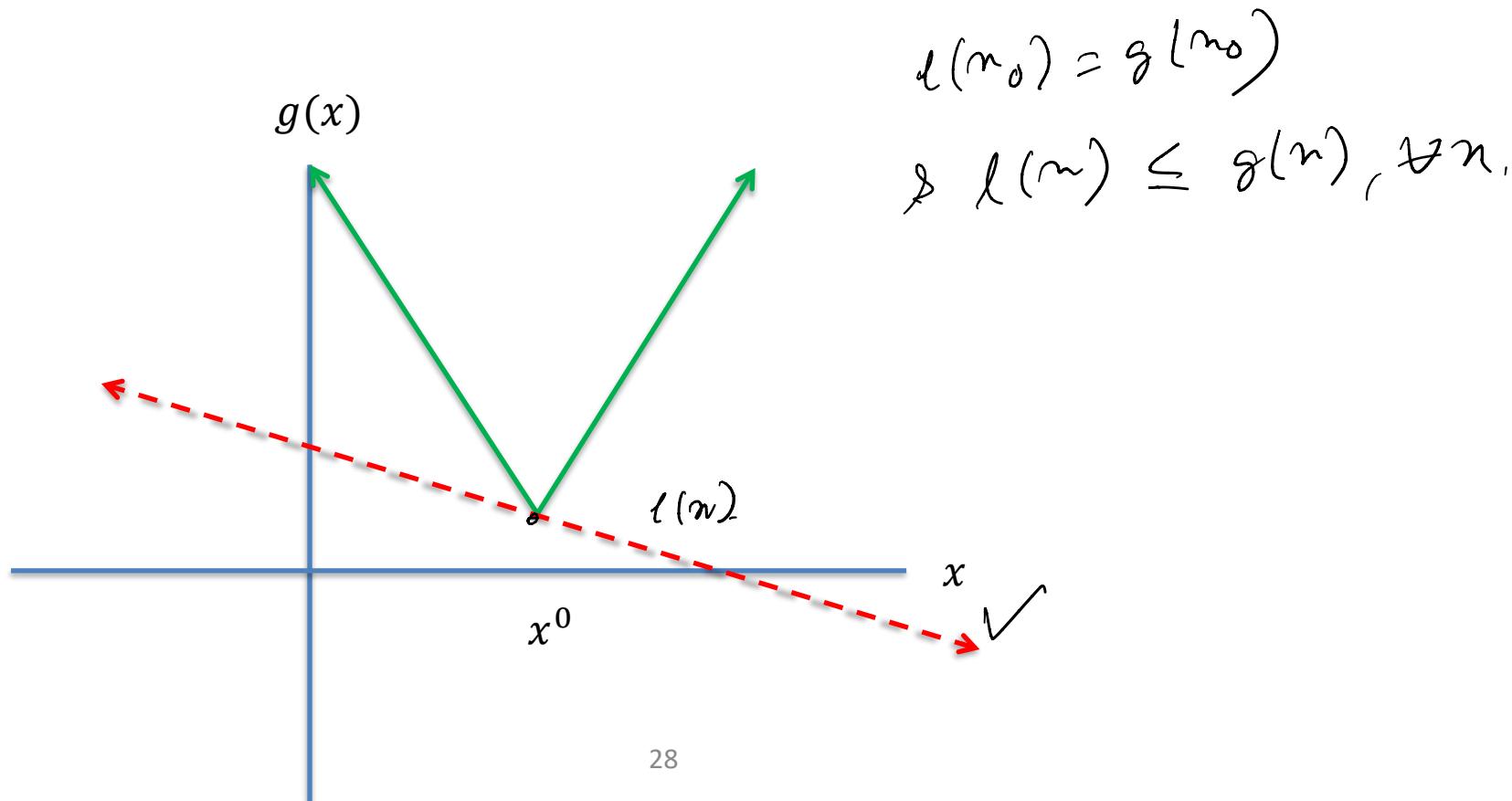
Subgradients

- For a convex function $g(x)$, a **subgradient** at a point x^0 is given by any line, l , such that $\underline{l(x^0) = g(x^0)}$ and $l(x) \leq g(x)$ for all x , i.e., it is a linear underestimator



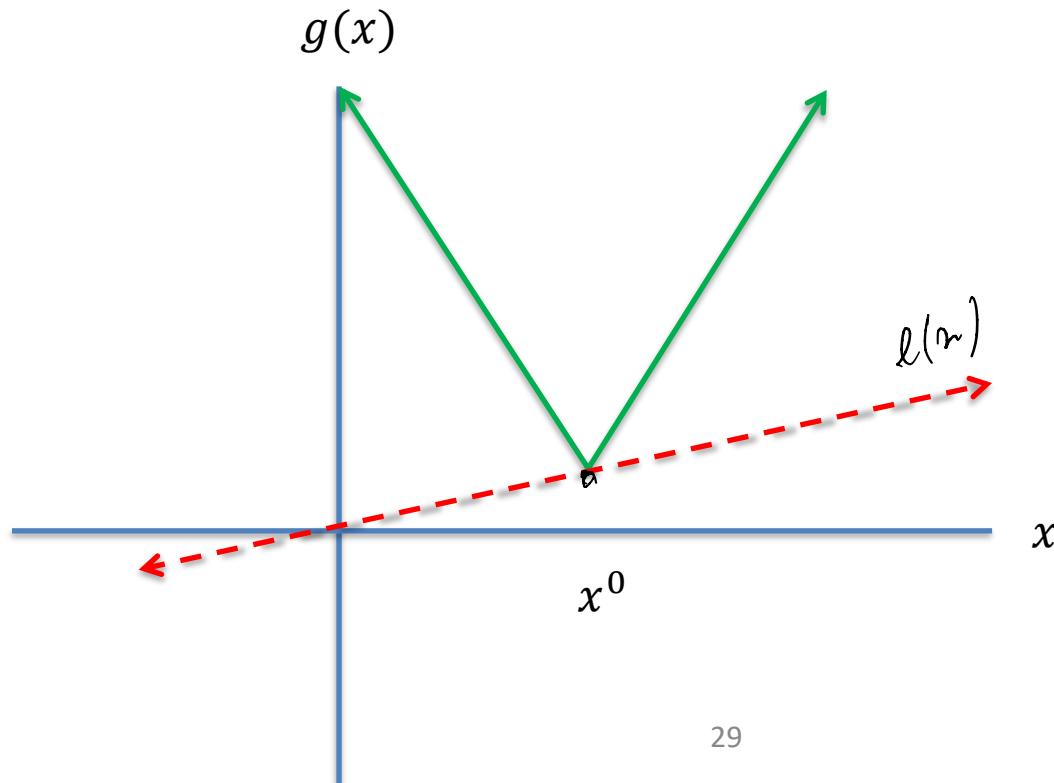
Subgradients

- For a convex function $g(x)$, a **subgradient** at a point x^0 is given by any line, l , such that $l(x^0) = g(x^0)$ and $l(x) \leq g(x)$ for all x , i.e., it is a linear underestimator



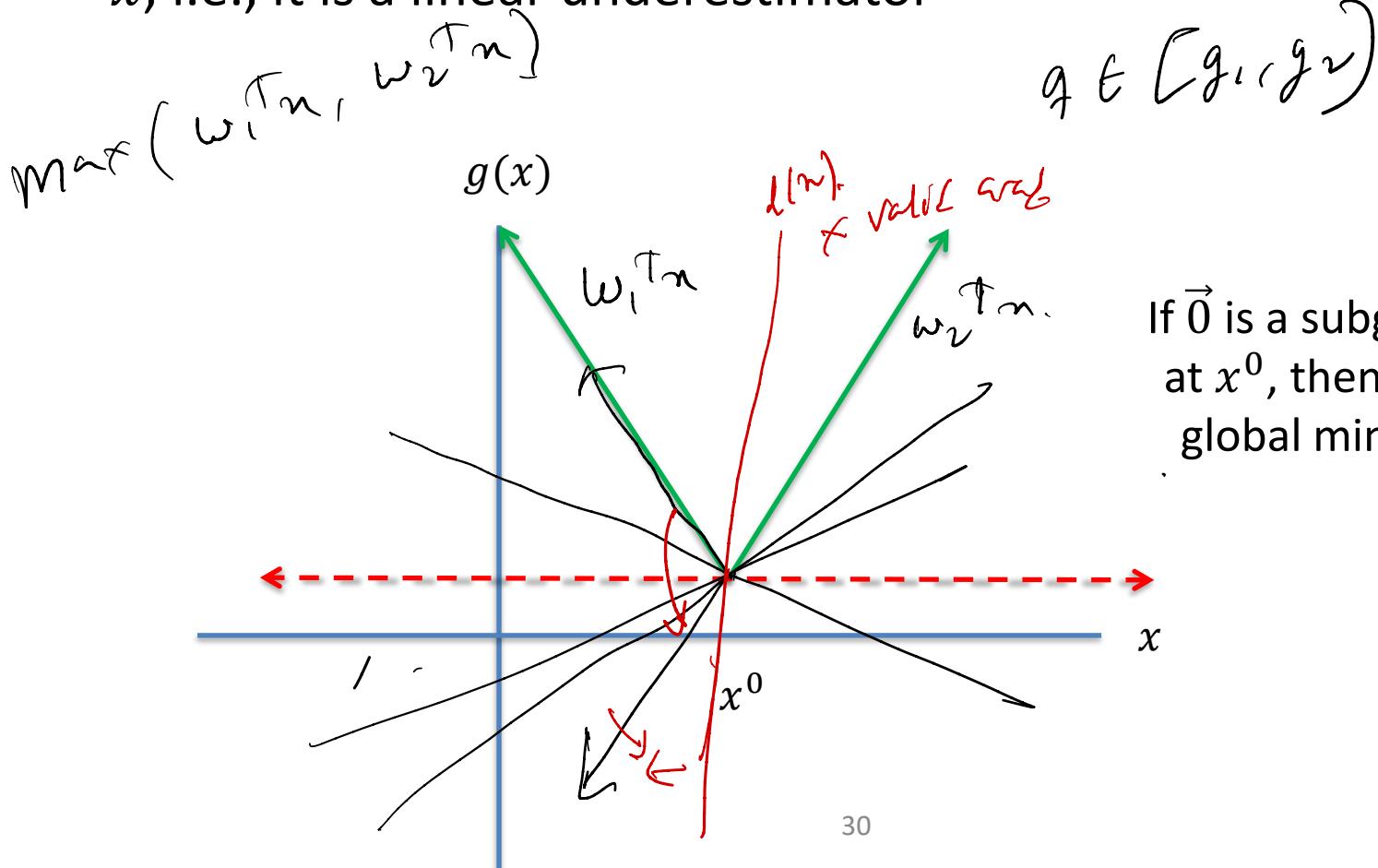
Subgradients

- For a convex function $g(x)$, a **subgradient** at a point x^0 is given by any line, l , such that $l(x^0) = g(x^0)$ and $l(x) \leq g(x)$ for all x , i.e., it is a linear underestimator



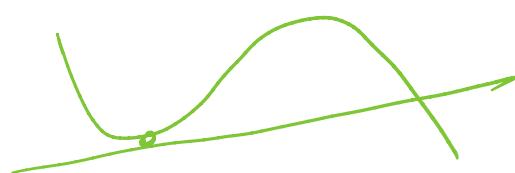
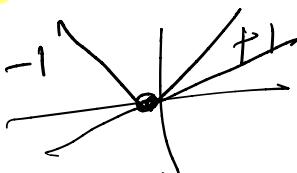
Subgradients

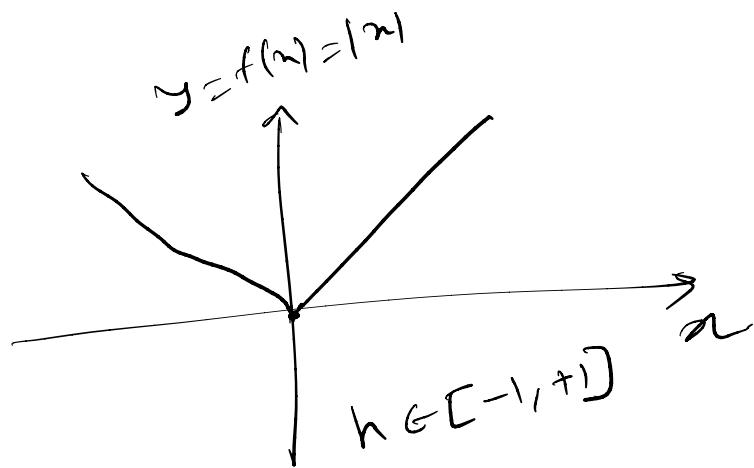
- For a convex function $g(x)$, a **subgradient** at a point x^0 is given by any line, l , such that $l(x^0) = g(x^0)$ and $l(x) \leq g(x)$ for all x , i.e., it is a linear underestimator



Subgradients

- If a convex function is differentiable at a point x , then it has a unique subgradient at the point x given by the gradient
- If a convex function is not differentiable at a point x , it can have many subgradients
 - E.g., the set of subgradients of the convex function $|x|$ at the point $x = 0$ is given by the set of slopes $[-1, 1]$
- Subgradients only guaranteed to exist for convex functions



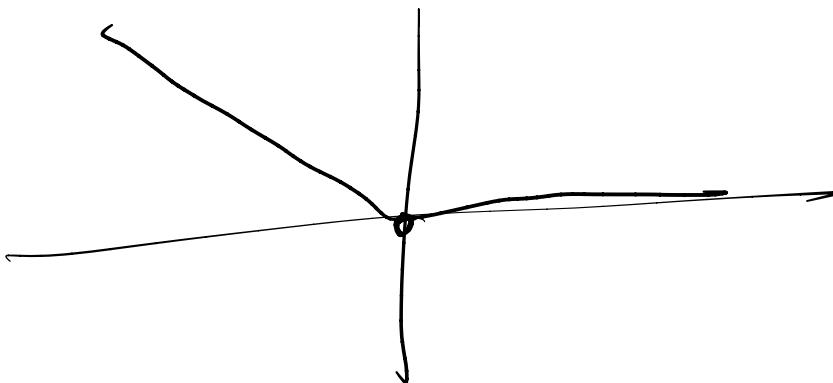


The Perceptron Algorithm

- Try to minimize the perceptron loss using (sub)gradient descent

$$\text{max} \left(0, -y^{(\omega^T n + b)} \right)$$

$$y^{(\omega^T n)} = 0$$



The Perceptron Algorithm

- Try to minimize the perceptron loss using (sub)gradient descent

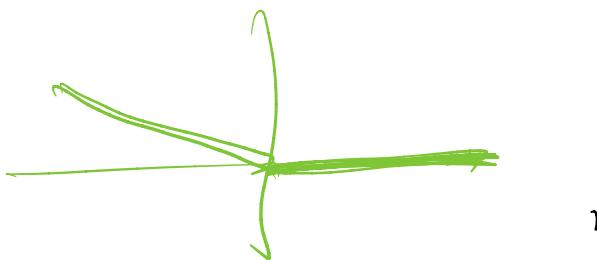
$$\nabla_w(\text{perceptron loss}) = - \sum_{m=1}^M \left(y^{(m)} x^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right)$$

$$L = f(g(\theta))$$

$$\nabla_b(\text{perceptron loss}) = - \sum_{m=1}^M \left(y^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right)$$

$$f(n) = \max(0, -n)$$

$$\begin{aligned} f'(n) &= -1 \downarrow n \leq 0 \\ &= -1 \downarrow n > 0 \end{aligned}$$



The Perceptron Algorithm

- Try to minimize the perceptron loss using (sub)gradient descent

$$\nabla_w (\text{perceptron loss}) = - \sum_{m=1}^M \left(y^{(m)} x^{(m)} \cdot 1_{-\underbrace{y^{(m)} f_{w,b}(x^{(m)})}_{\geq 0}} \right)$$

$$\nabla_b (\text{perceptron loss}) = - \sum_{m=1}^M \left(y^{(m)} \cdot 1_{-\underbrace{y^{(m)} f_{w,b}(x^{(m)})}_{\geq 0}} \right)$$

if $y^{(m)}$ & $f_{w,b}(x^{(m)})$ have
same sign.

$$f_{w,b}(x) = w^T x + b$$

Is equal to zero if the m^{th} data point is correctly classified and one otherwise

The Perceptron Algorithm

- Try to minimize the perceptron loss using (sub)gradient descent

Full gradient

$$w^{(t+1)} = w^{(t)} + \gamma_t \sum_{m=1}^M \left(y^{(m)} x^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right) \nabla_w L$$

$$b^{(t+1)} = b^{(t)} + \gamma_t \sum_{m=1}^M \left(y^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right) \nabla_b L$$

- With step size γ_t (also called the learning rate)
- Note that, for convergence of subgradient methods, a diminishing step size, e.g., $\gamma_t = \frac{1}{1+t}$ is required

Stochastic Gradient Descent

- To make the training more practical, **stochastic (sub)gradient descent** is often used instead of standard gradient descent
- Approximate the gradient of a sum by sampling a few indices (as few as one) uniformly at random and averaging

$$\nabla_x \left[\sum_{m=1}^M g_m(x) \right] \approx \frac{1}{K} \sum_{k=1}^K \nabla_x g_{m_k}(x)$$

here, each m_k is sampled uniformly at random from $\{1, \dots, M\}$

- Stochastic gradient descent converges to the global optimum under certain assumptions on the step size

Stochastic Gradient Descent



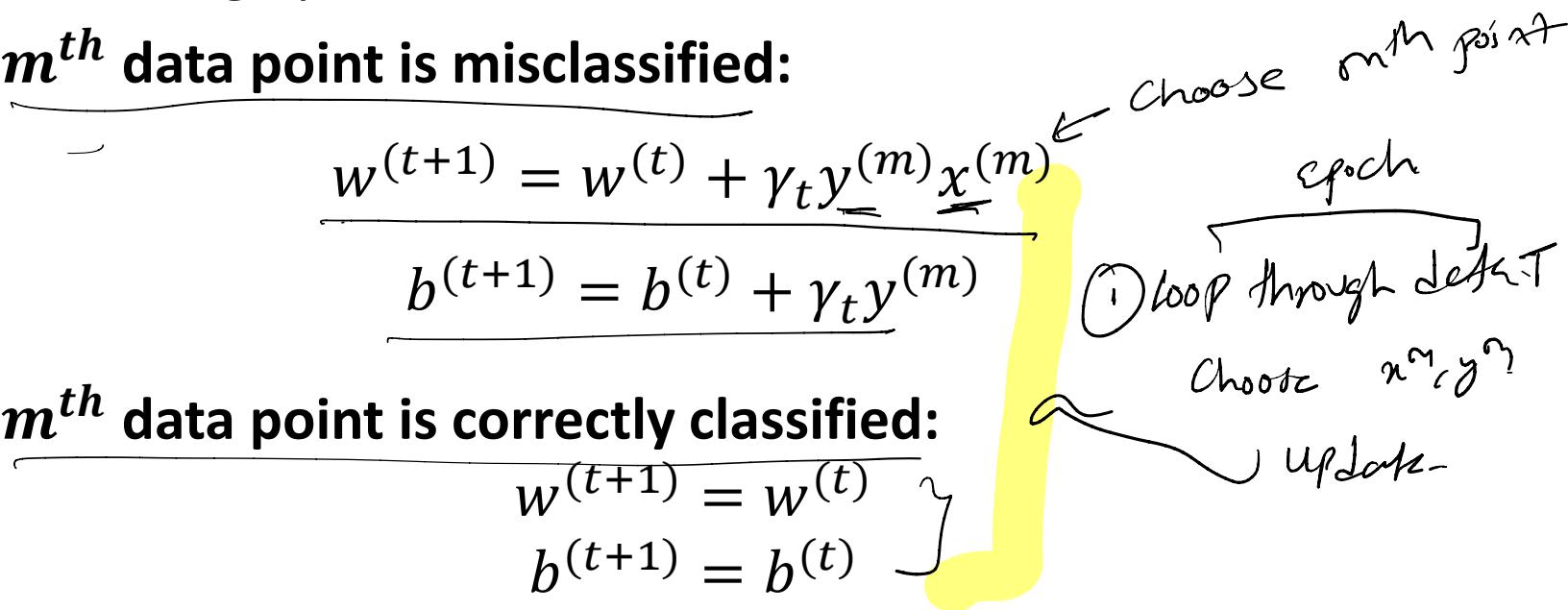
- Setting $K = 1$, we pick a random observation m and perform the following update

if the m^{th} data point is misclassified:

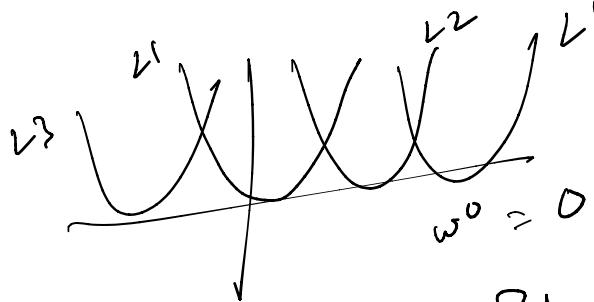
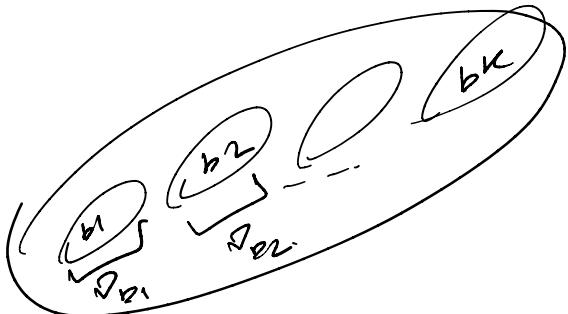
$$\begin{aligned} w^{(t+1)} &= w^{(t)} + \gamma_t y^{(m)} \underline{x}^{(m)} \\ b^{(t+1)} &= b^{(t)} + \gamma_t y^{(m)} \end{aligned}$$

if the m^{th} data point is correctly classified:

$$\begin{aligned} w^{(t+1)} &= w^{(t)} \\ b^{(t+1)} &= b^{(t)} \end{aligned}$$



- Sometimes, you will see the perceptron algorithm specified with $\gamma_t = 1$ for all t



$$L = L_1 + L_2 + L_3 + \dots$$

~~Burk
Noise~~

UP: $\nabla L = \sum_{i=1}^n \nabla L_i$ & ^{Loop through} _{derivative}

$$w' = w^0 - \eta \nabla L(w^0)$$

4 step

SDP

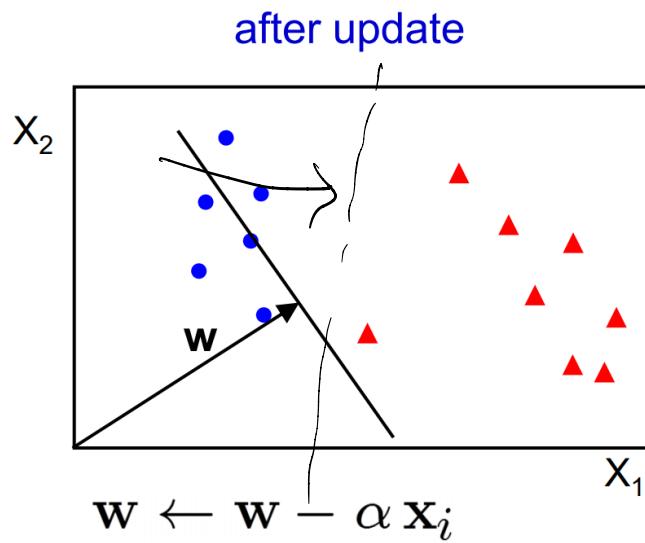
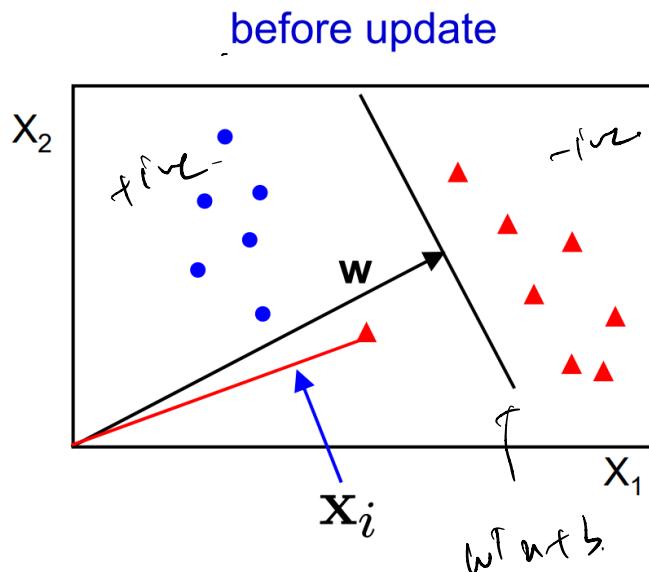
$$w^0 = 0$$

$$\begin{cases} w^1 = w^0 - \eta \nabla L_1(w^0) \\ w^2 = w^1 - \eta \nabla L_2(w^1) \\ w^3 = w^2 - \eta \nabla L_3(w^2) \\ w^4 = w^3 - \eta \nabla L_4(w^3) \end{cases}$$

Perceptron Example

For example in 2D

- Initialize $\mathbf{w} = 0$
- Cycle though the data points $\{ \mathbf{x}_i, y_i \}$
 - if \mathbf{x}_i is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \text{sign}(f(\mathbf{x}_i)) \mathbf{x}_i$
- Until all the data is correctly classified

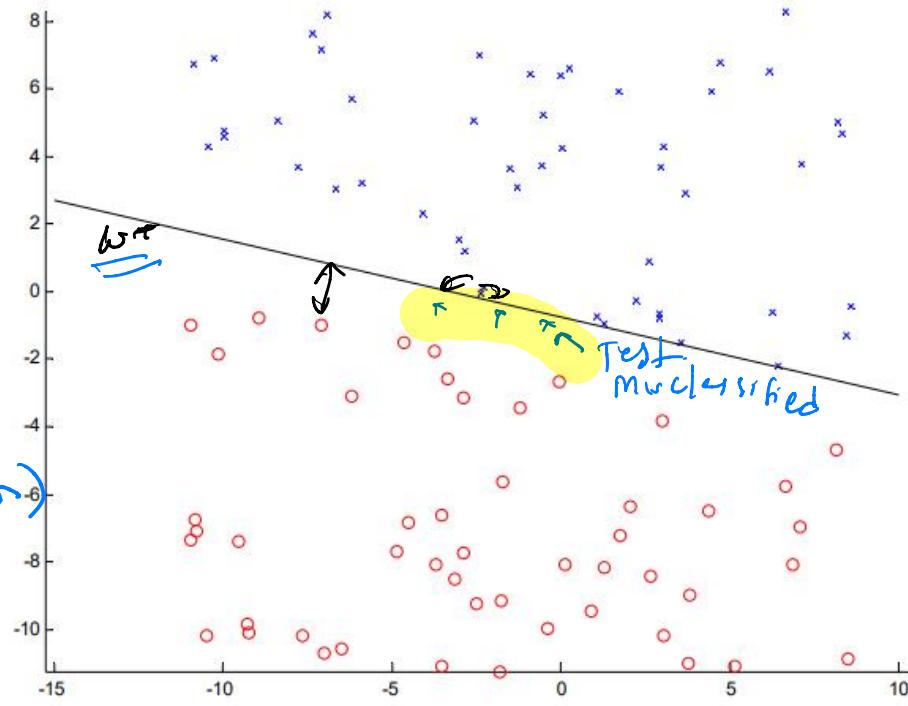
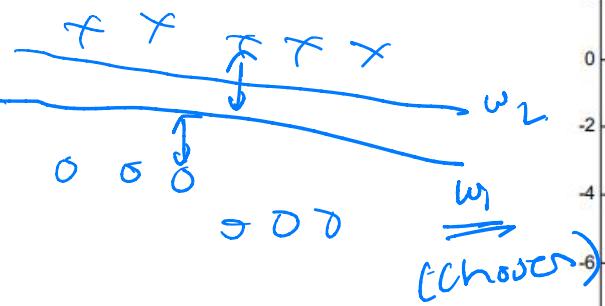


NB after convergence $\mathbf{w} = \sum_i^N \alpha_i \mathbf{x}_i$

Part III: More On Perceptron

More on Perceptron

Perceptron example



① if lin sep.

$$ML = 0$$

$$\Rightarrow PL = 0$$

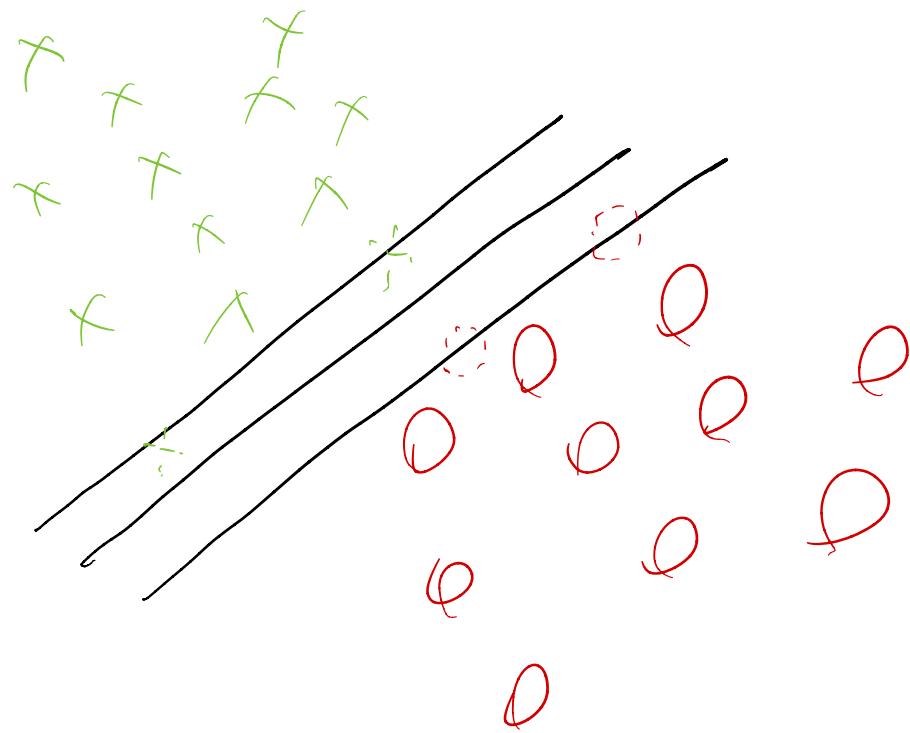
∇w^* st

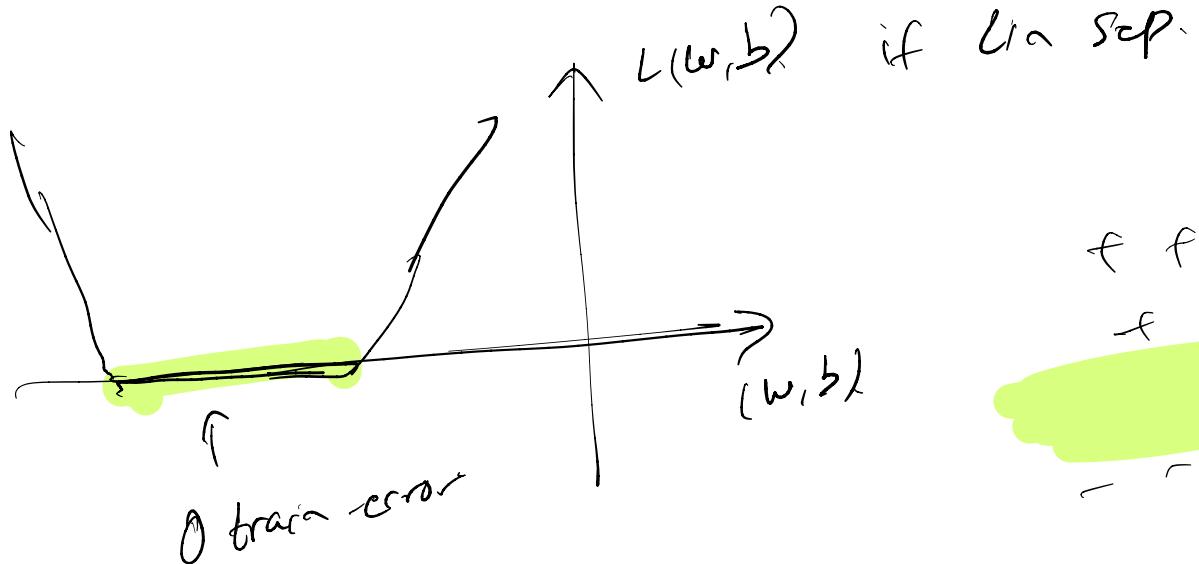
$$PL(w^*) = 0$$

② PL is convex

so SGD will converge to w^*

- if the data is linearly separable, then the algorithm will converge
- convergence can be slow ...
- separating line close to training data ↪ $B \leftarrow$ for less
- we would prefer a larger margin for generalization





Applications of Perceptron

- Spam email classification

Bags of words model

- Represent emails as vectors of counts of certain words (e.g., *sir, madam, Nigerian, prince, money*, etc.)

Rishabh

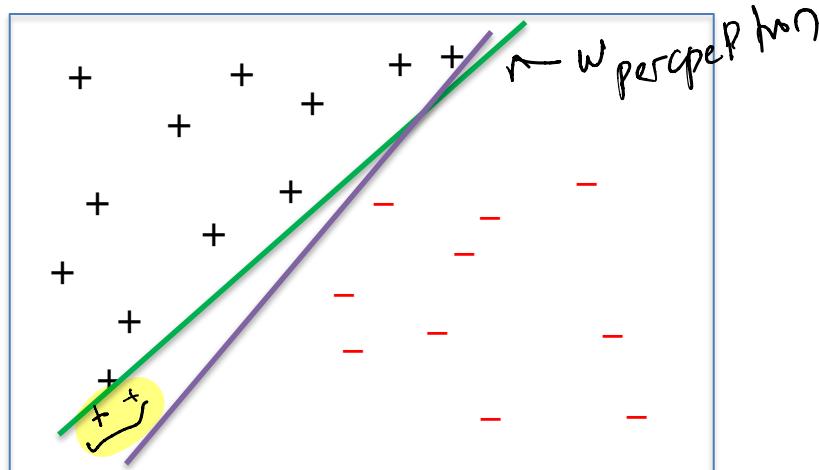
- Apply the perceptron algorithm to the resulting vectors
- To predict the label of an unseen email

- Construct its vector representation, $x' \leftarrow$ test email.

- Check whether or not $w^T x' + b$ is positive or negative

Perceptron Learning Drawbacks

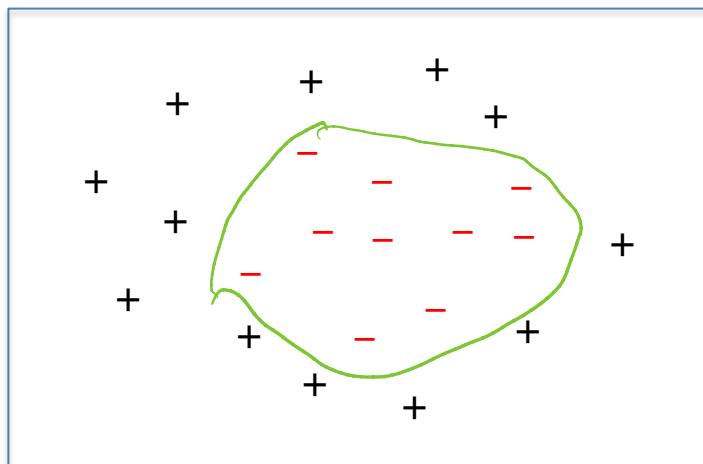
- No convergence guarantees if the observations are not linearly separable
- Can overfit
 - There can be a number of perfect classifiers, but the perceptron algorithm doesn't have any mechanism for choosing between them



What If the Data Isn't Separable?

↓
linearly

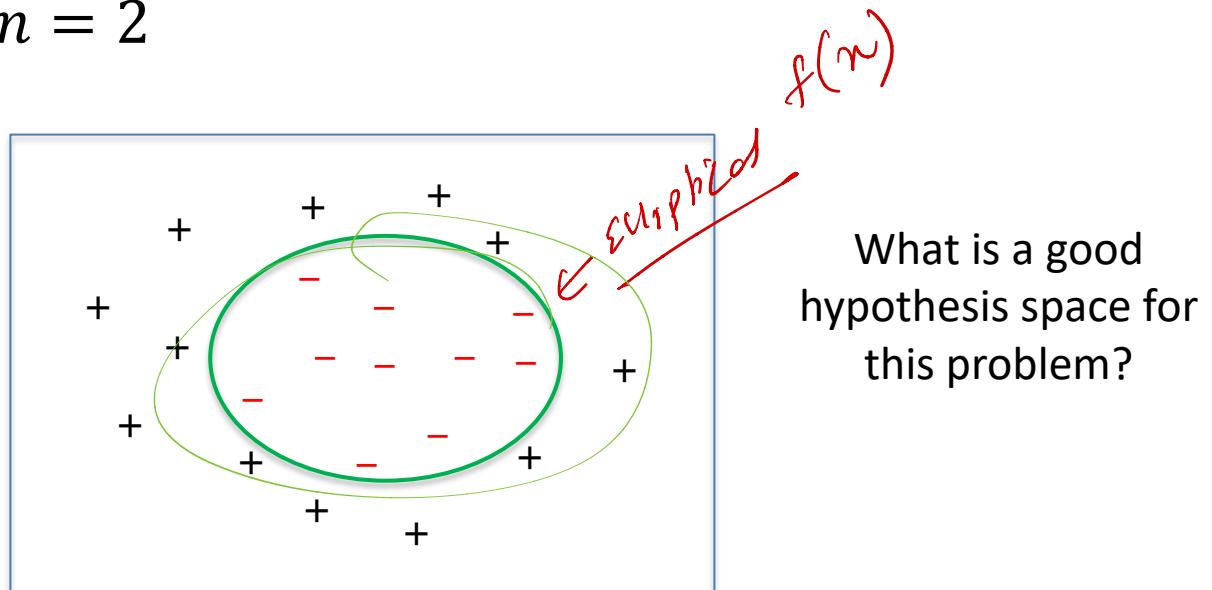
- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- We can think of the observations as points in \mathbb{R}^n with an associated sign (either +/- corresponding to 0/1)
- An example with $n = 2$



What is a good
hypothesis space for
this problem?

What If the Data Isn't Separable?

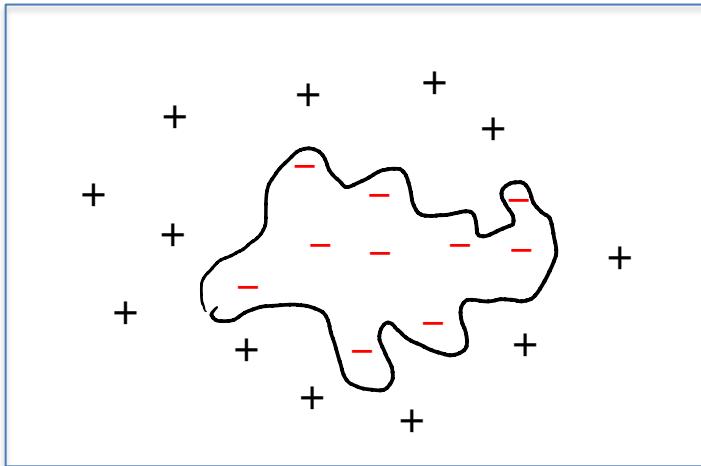
- Input $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- We can think of the observations as points in \mathbb{R}^n with an associated sign (either +/- corresponding to 0/1)
- An example with $n = 2$



Adding Features

- Perceptron algorithm only works for linearly separable data

n



Can add **features** to make the data linearly separable in a
higher dimensional space!

Essentially the same as higher order polynomials for linear
regression!

Adding Features

- The idea, choose a feature map $\phi: \mathbb{R}^n \rightarrow \mathbb{R}^k$

x_1, \dots, x_n
 $\underbrace{\quad\quad\quad}_{n \text{ features}}$

- Given the observations $x^{(1)}, \dots, x^{(M)}$, construct feature vectors $\phi(x^{(1)}), \dots, \phi(x^{(M)})$
- Use $\phi(x^{(1)}), \dots, \phi(x^{(M)})$ instead of $x^{(1)}, \dots, x^{(M)}$ in the learning algorithm
- Goal is to choose ϕ so that $\phi(x^{(1)}), \dots, \phi(x^{(M)})$ are linearly separable in \mathbb{R}^k
- Learn linear separators of the form $w^T \underline{\phi(x)}$ (instead of $w^T x$)
- Warning:** more expressive features can lead to overfitting!

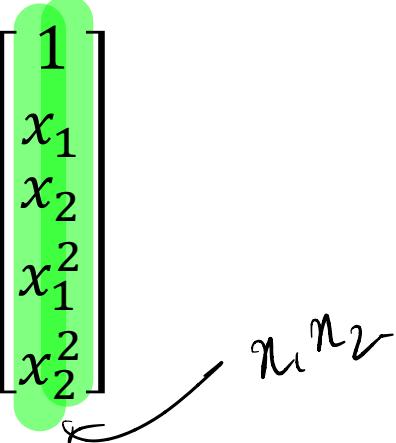
Adding Features: Examples

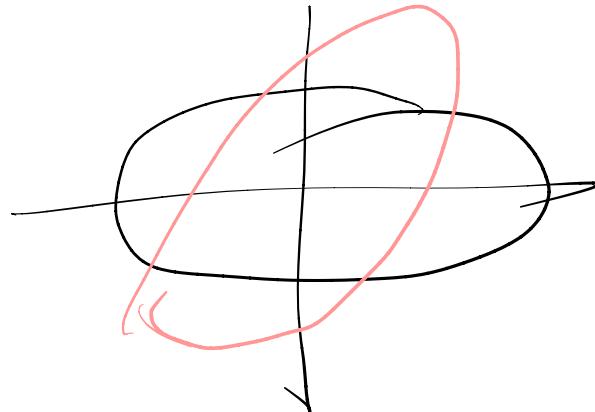
$$\bullet \phi \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- This is just the input data, without modification

$$\bullet \phi \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

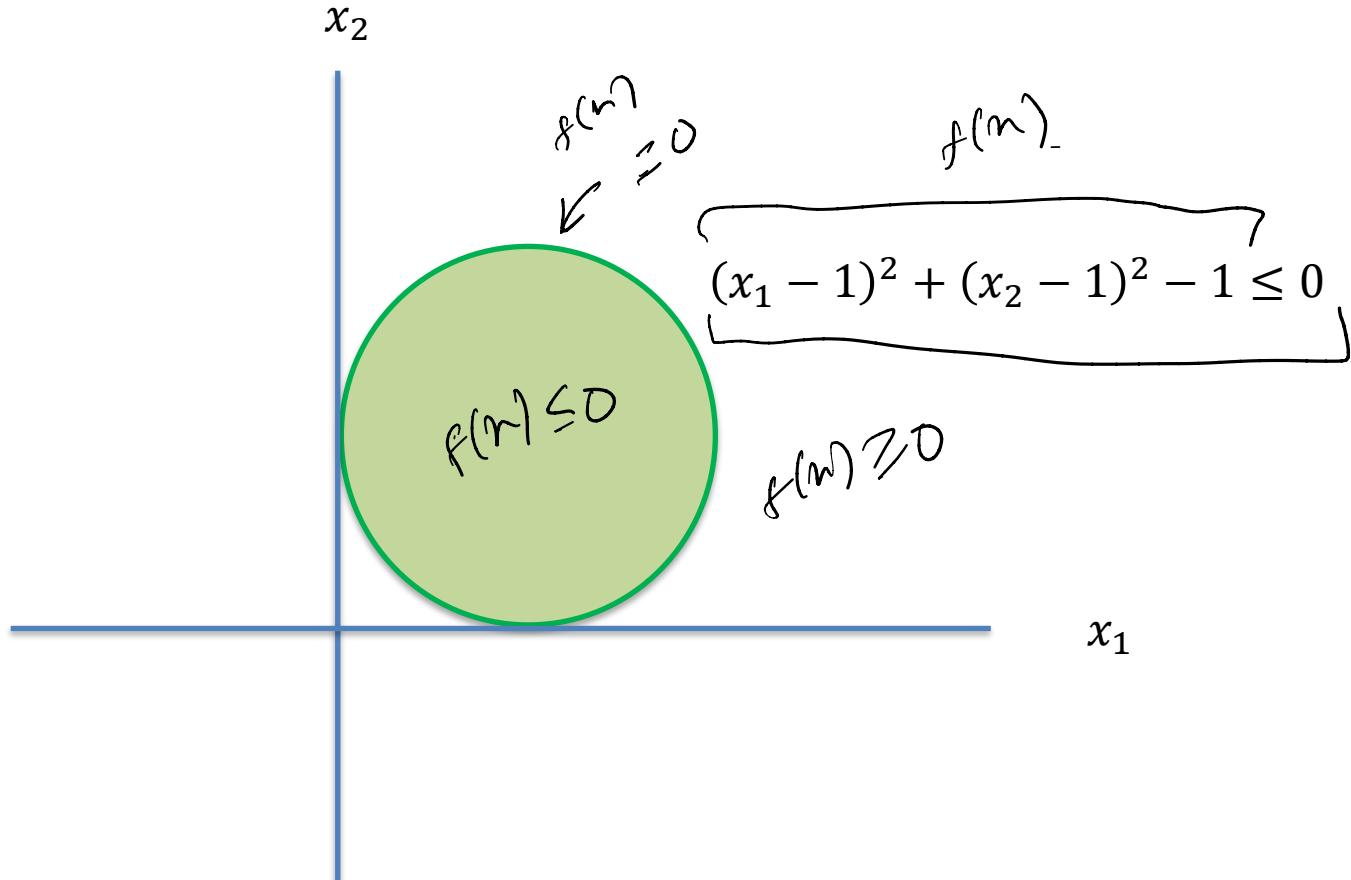
n n^2



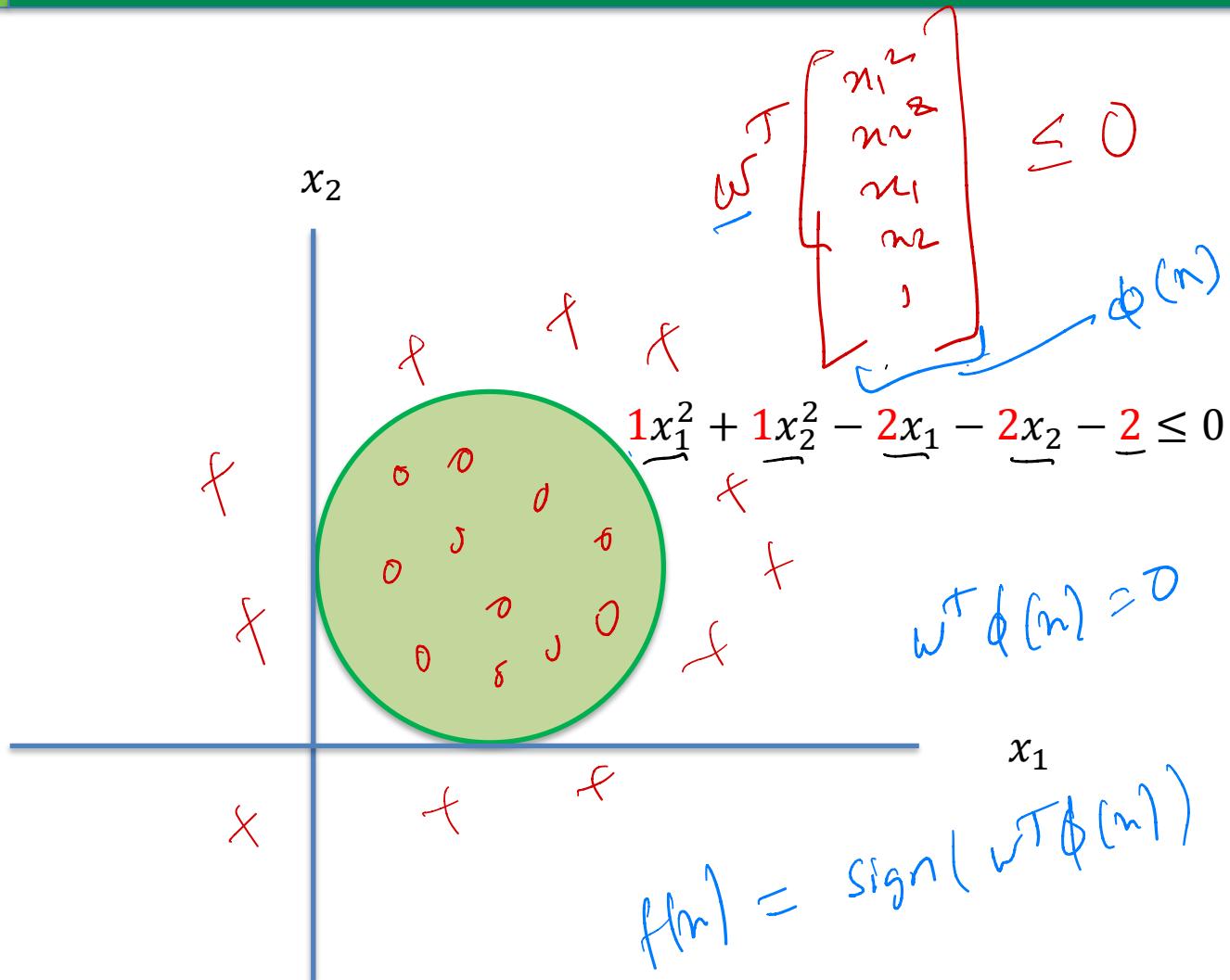


- This corresponds to a **second-degree polynomial** separator, or equivalently, **elliptical separators** in the original space

Adding Features



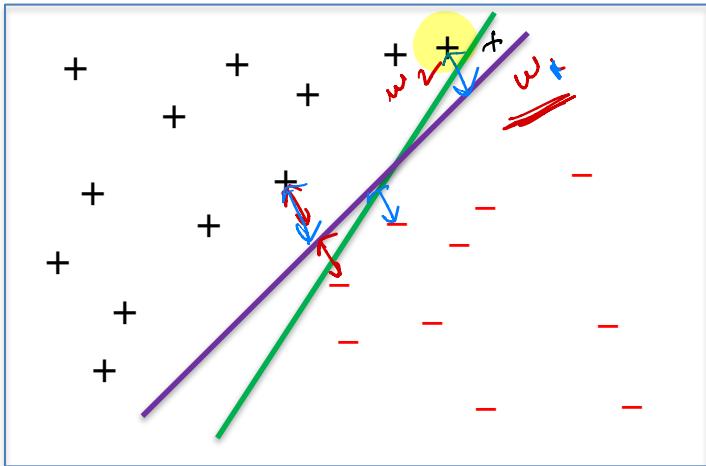
Adding Features



Support Vector Machines



- How can we decide between two perfect classifiers?



w_1 = Equal margin
between classes

w_2 = Close to one
class

↓
Prone to
overfitting

- What is the practical difference between these two solutions?