COGS 118A, Winter 2020

Supervised Machine Learning Algorithms

Lecture 11: Classifier Complexity

Zhuowen Tu

Midterm II

Midterm II, 02/27/2020 (Thursday)

Time: 12:30-13:50PM

Location: Ledden Auditorium

You can bring one page "cheat sheet". No use of computers/smart-phones during the exam.

Bring your pen.

Bring your calculator.

A study guide and practice questions will be provided.

Linear Model

A linear model:

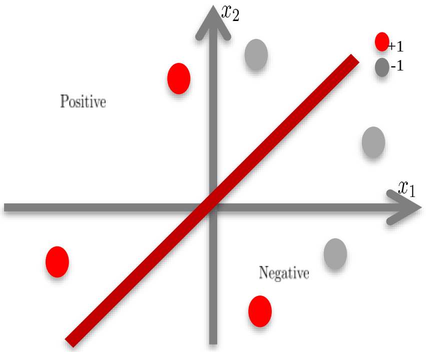
$$f(\mathbf{x};\mathbf{w},b) =< \mathbf{w},\mathbf{x} > +b$$
$$= \mathbf{w}\cdot\mathbf{x} + b$$
$$= \mathbf{w}^T\mathbf{x} + b$$

$\mathbf{x} = \mathbb{R}^m$ $\mathbf{w} = \mathbb{R}^m$ $b \in \mathbb{R}$

This is a linear function and our job is find the optimal **w** and b to best fit the prediction in learning.

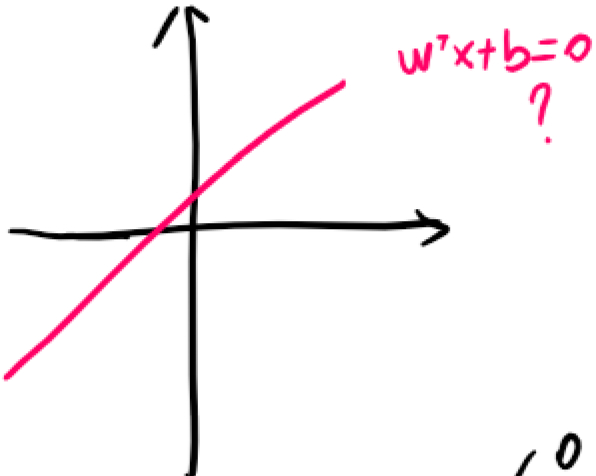
A summary of
the classifiers so far

$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$



- Perceptron
- Logistic regression classifier
- Support vector machine (SVM)

The decision stump classifier is also a linear classifier



Decision Stump:

$$f(x; j, th) = \begin{cases} +1 & \text{if } x_j \geq th \\ -1 & \text{else} \end{cases}$$

$$w = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \rightarrow j \qquad b = -th$$

Main motivation

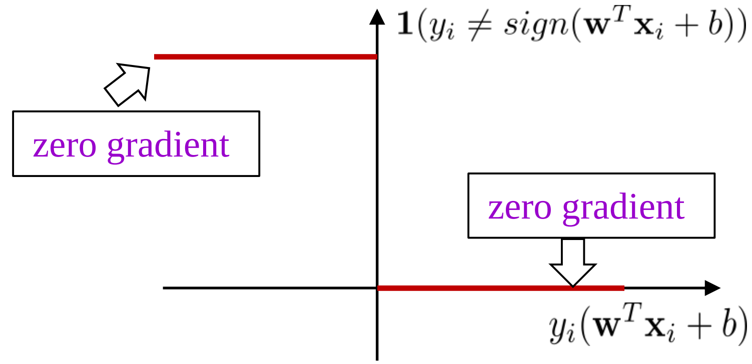
Hard->Half-hard->Soft

Error

Standard loss (error) function

Standard 0/1 loss (gradient 0 nearly everywhere, no gradient feedback):

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i \mathbf{1}(y_i \neq sign(\mathbf{w}^T \mathbf{x}_i + b))$



It is the most directly loss, but is also the hardest to minimize.

Zero gradient everywhere!

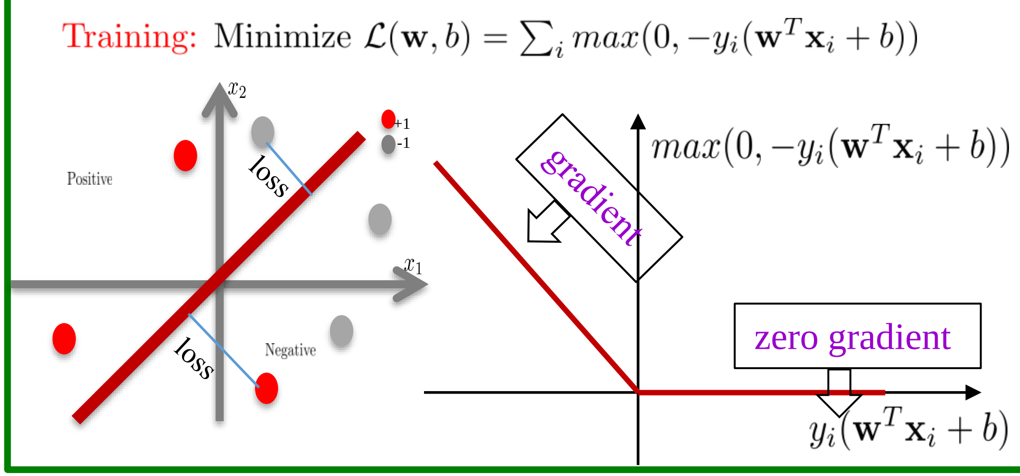
Main motivation

Hard->Half-hard->Soft

Error

Half-hard loss (error) function

Loss implicitly used in the perceptron algorithm: with gradient feedback when the target (ground-truth label) and the output (classification) are different).



Zero loss for correct classification (no gradient).

A loss based on the distance to the decision boundary for misclassification (with gradient).

Used in the perceptron training.

Soft loss (error) function

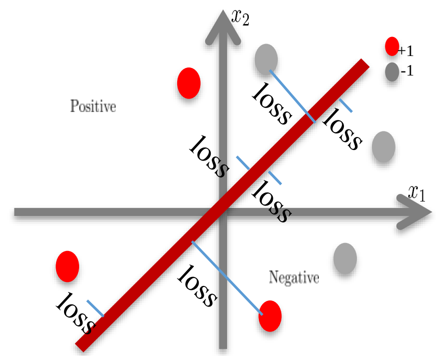
Main motivation

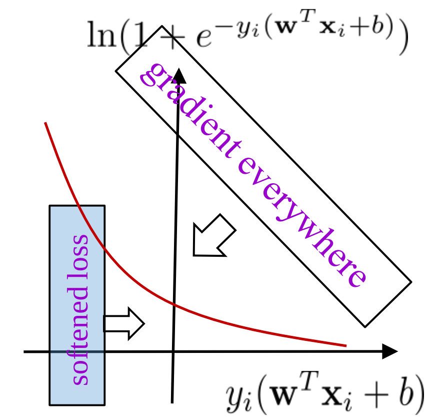
Hard->Half-hard->Soft

Error

Loss used in logistic regression.

Training: minimize $\mathcal{L}(\mathbf{w}, b) = \sum_{i=1}^{n} \ln(1 + e^{-y_i(\mathbf{w}^T\mathbf{x}_i + b)})$





Every data point receives a loss (gradient everywhere).

A loss based on the distance to the decision boundary for wrong classification (has a gradient).

Used in logistic regression classifier.

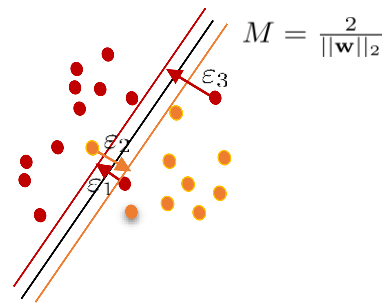
Loss in SVM

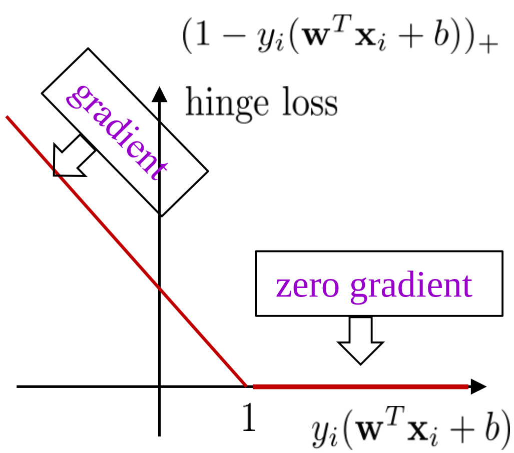
Main motivation

Hard->Hinge

Error

Minimize $\mathcal{L}(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+$

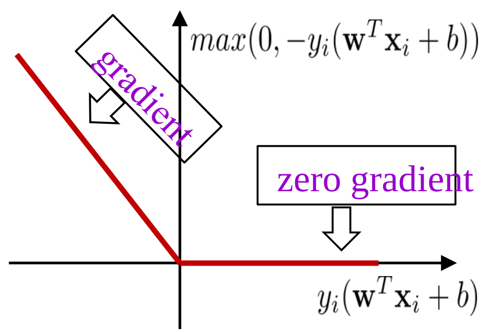
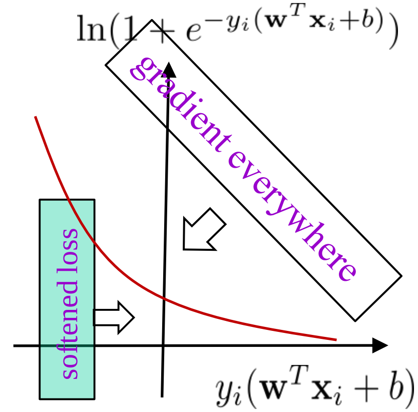
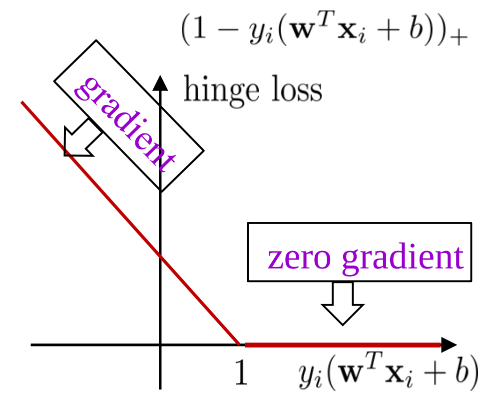
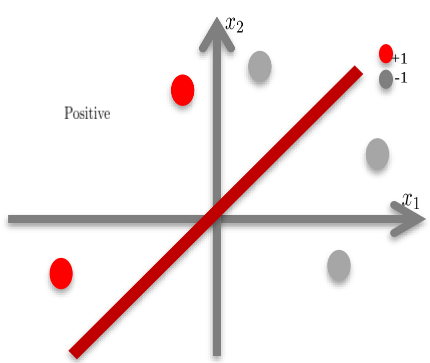
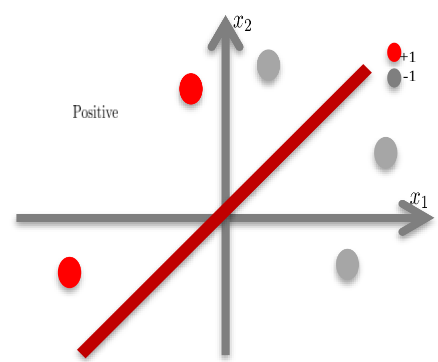
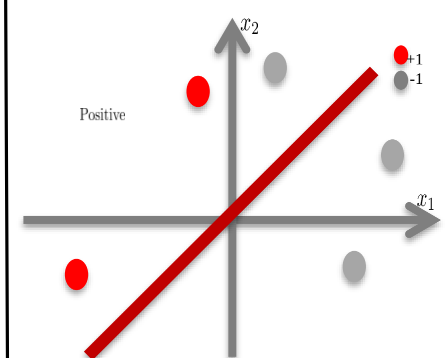




Zero loss for correct classification beyond the margin (no gradient).

A loss based on the distance to the decision boundary for misclassification or within the margin (with gradient).

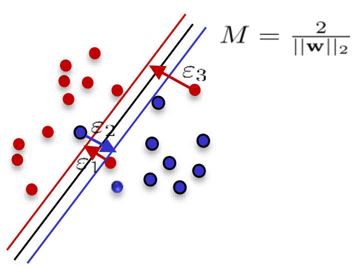
A Summary

	Perceptron	Logistic Regression	SVM
Training	Training: Minimize $\mathcal{L}(\mathbf{w},b)=\sum_i max(0,-y_i(\mathbf{w}^T\mathbf{x}_i+b))$	Training: Minimize $\mathcal{L}(\mathbf{w},b)=\sum_{i=1}^{n}\ln(1+e^{-y_i(\mathbf{w}^T\mathbf{x}_i+b)})$	Training: Minimize $\mathcal{L}(\mathbf{w},b)=\frac{1}{2}\lvert\lvert\mathbf{w}\rvert\rvert^2 +C\sum_{i=1}^{n}(1-y_i(\mathbf{w}^T\mathbf{x}_i+b))_+$
			
	convex optimization	convex optimization	convex optimization
Testing			
	Test: $f(\mathbf{x};\mathbf{w},b)=\begin{cases}+1 & if\ \mathbf{w}^T\mathbf{x}+b\geq 0\\ -1 & otherwise\end{cases}$	Test: $f(\mathbf{x};\mathbf{w},b)=\begin{cases}+1 & if\ \mathbf{w}^T\mathbf{x}+b\geq 0\\ -1 & otherwise\end{cases}$	Test: $f(\mathbf{x};\mathbf{w},b)=\begin{cases}+1 & if\ \mathbf{w}^T\mathbf{x}+b\geq 0\\ -1 & otherwise\end{cases}$

Recap: Support Vector Machine

Intuition: It explicitly introduces a "regularization" (margin) into the objective function to combine with a classification error (restricted using a hinge loss) term.

- It achieves unprecedented robustness when training a linear classifier due to the use of margin term in training.
- The learned model is based on a balance between classification error and margin. The balancing term C is typically attained using cross-validation.
- Kernel based SVM makes non-separable samples feasible to classify by projecting the data onto higher dimensional spaces.
- The features defined under kernels don't need to be computed explicitly.
- The learned weights $\mathbf{w}$ is carried in the weights for the samples and those samples with non-zero weights are called support vectors.



What's the difference between logistic regression and linear SVM?

Logistic regression:

$$p(y|\mathbf{x}) = \frac{1}{1+e^{-y(\mathbf{w}^T\mathbf{x}+b)}}$$

Training: $\arg\min_{(\mathbf{w},b)} \sum_{i=1}^{n} \ln(1 + e^{-y_i(\mathbf{w}^T\mathbf{x}_i+b)})$

Test: $f(\mathbf{x};\mathbf{w},b) = \begin{cases} +1 & if\ \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+b)}} \geq 0.5 \\ -1 & otherwise \end{cases}$

Equivalent to: $f(\mathbf{x};\mathbf{w},b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x}+b \geq 0 \\ -1 & otherwise \end{cases}$

SVM:

Training: $\arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+$

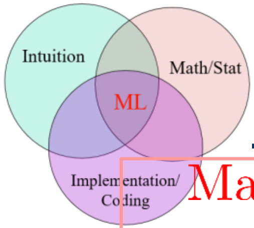
Test: $f(\mathbf{x};\mathbf{w},b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x}+b \geq 0 \\ -1 & otherwise \end{cases}$

A. They are different in both training and testing.

B. They differ in training but are the same in testing.

C. They differ in testing but are the same in training.

D. They are completely different.



Recap: Support Vector Machine

Math:

Training :

$$\text{Minimize } \mathcal{L}(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+$$

$$\frac{\mathcal{L}(\mathbf{w},b)}{\partial \mathbf{w}} = \mathbf{w} + C\sum_{i=1}^{n}\begin{cases} 0 & if \;\; y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \\ -y_i\mathbf{x}_i & otherwise \end{cases}$$

$$\frac{\mathcal{L}(\mathbf{w},b)}{\partial b} = C\sum_{i=1}^{n}\begin{cases} 0 & if \;\; y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \\ -y_i & otherwise \end{cases}$$

Testing :

$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if \; \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$

Implementation:

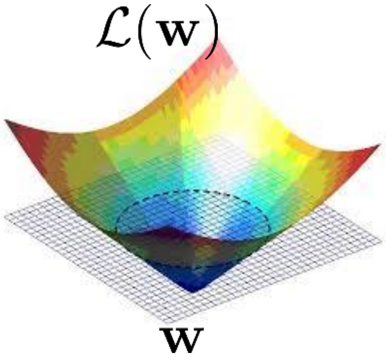
Gradient Descent Direction

(a) Pick a direction $\nabla\mathcal{L}(\mathbf{w}_t, b_t)$

(b) Pick a step size λ_t

(c) $\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda_t \times \nabla\mathcal{L}_{\mathbf{w}_t}(\mathbf{w}_t, b_t)$ such that function decreases;
$b_{t+1} = b_t - \lambda_t \times \nabla\mathcal{L}_{b_t}(\mathbf{w}_t, b_t)$

(d) Repeat



Classifier Complexity

Summary of the classification problem

$\mathbf{x} = (x_1, x_2, \dots)$

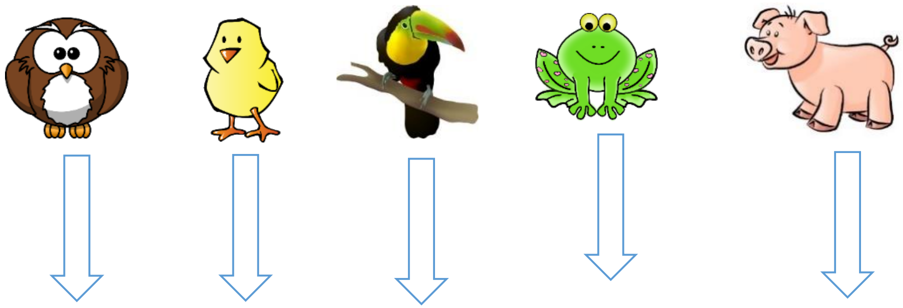
$y = 1(bird)$

x_1: color

x_2 weight

$\dots$

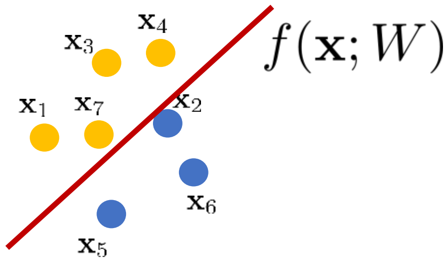
$$S_{training} =$$



$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4), (\mathbf{x}_5, y_5)\}$$

Train classifier $f(\mathbf{x}; W)$

W: model parameter

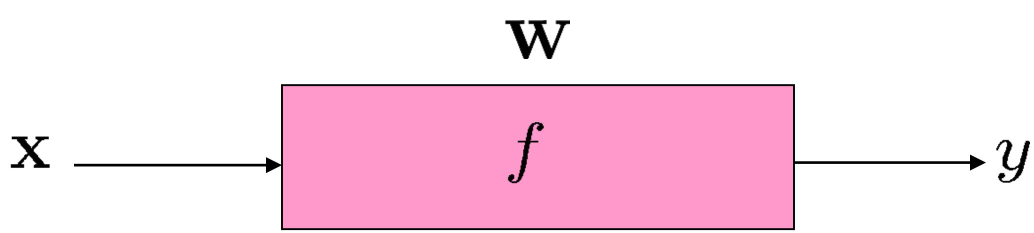


VC Dimension

This leads to the theoretic analysis about the VC dimension theory by Vapnik (Vapnik 1982).

A learning machine **f** takes an input **x** and transforms it, somehow using weights α, into a predicted output

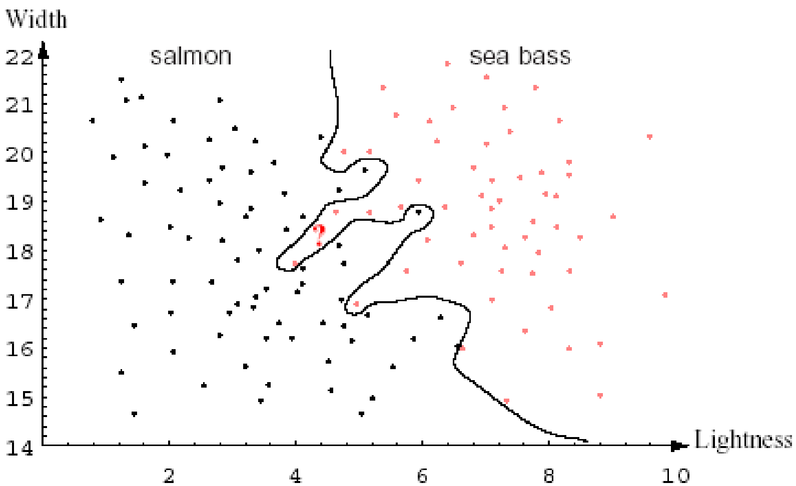
$\mathbf{w}$ is some vector of adjustable parameters.

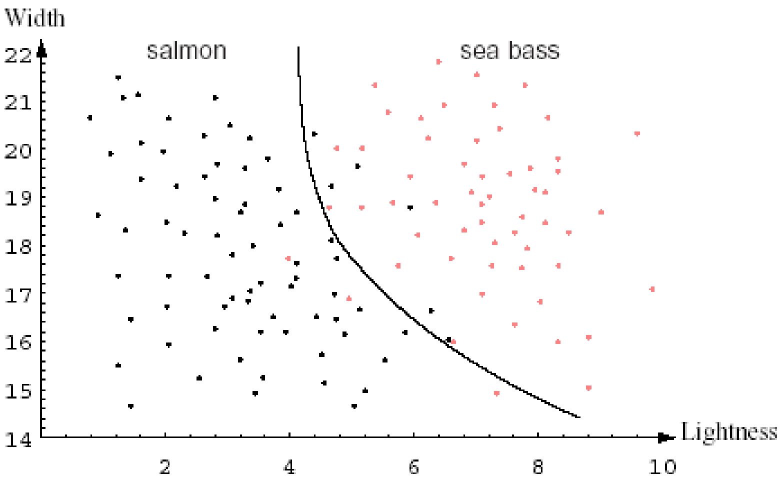


$$y \in \{-1, +1\}$$

Error

Now, let $f(\mathbf{x};\ \mathbf{W})$ be one classifier which makes the prediction for the label y, we define the error on a set of input as:





$$S_{testing} = \{(\mathbf{x}_i, y_i), i = 1..q\}$$

$$e_{testing} = \frac{1}{q} \sum_{i=1}^{q} \mathbf{1}(y_i \neq f(\mathbf{x}_i; W))$$

$$\mathbf{1}(z) = \begin{cases} 1 & if\ z = TRUE \\ 0 & otherwise \end{cases}$$

Vapnik-Chervonenkis Dimension

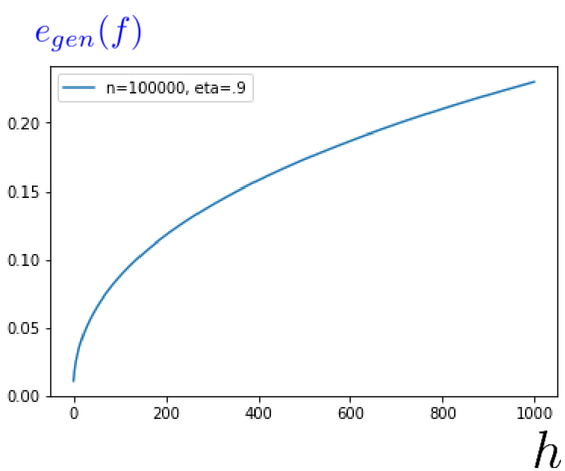
$$Pr(e_{testing} \leq e_{training} + e_{gen}(f)) = 1 - \eta$$

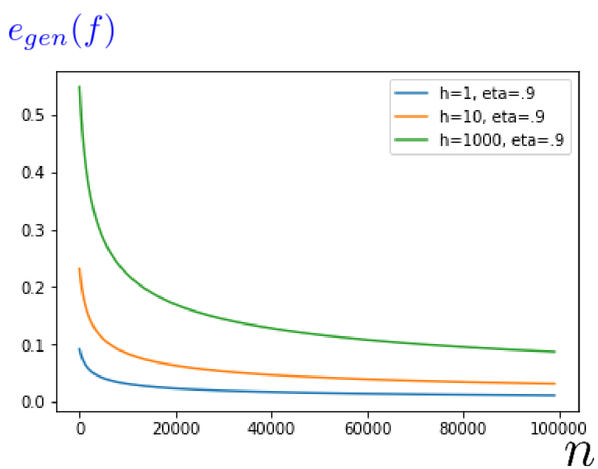
$$e_{gen}(f) = \sqrt{\frac{h(\log(2n/h+1)) - \log(\eta/4)}{n}}$$

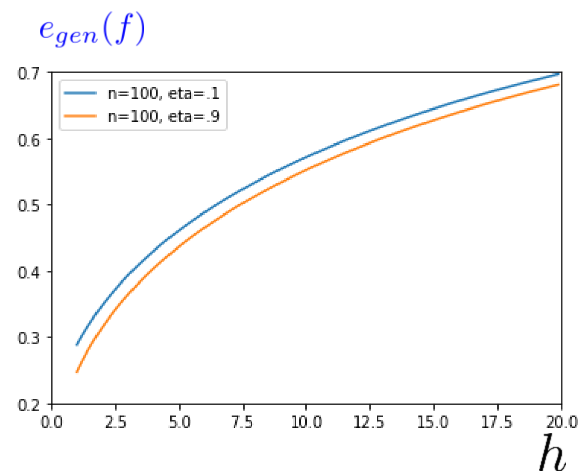
n: the number of training samples

h: complexity (VC dimension) of the classifier f

$1 - \eta$: confidence







Vapnik-Chervonenkis Dimension

$$e_{testing} \leq e_{training} + \sqrt{\frac{h(\log(2n/h+1)) - \log(\eta/4)}{n}}$$

- This gives us a way to estimate the error on future data based only on the training error and the VC-dimension (h) of **f**.
- This is a not so tight bound and it makes sense when n is big (e.g. > 10,000).
- In order to reduce the testing error, we need to reduce both the training error and the VC dimension of the classifier.
- h is the VC dimension but how do we compute h?

n: the number of training samples

η: confidence level, can be ignored for the moment

VC-dimension

Theory: The VC dimension of the set of oriented hyperplanes in $\mathbb{R}^r$ is $r+1$, since we can always choose r+1 points, and then choose one of the points as origin, such that the position vectors of the remaining r points are linearly independent, but can never choose r+2 such points.

For a linear classifier:

$$f(\mathbf{x}; \mathbf{w}, b) = sign(\mathbf{w}^T \mathbf{x} + b),$$

$$\mathbf{x}, \mathbf{w} \in \mathbb{R}^r, \; b \in \mathbb{R}$$

Total number of parameters: $r + 1$.

What does VC-dimension (h) measure?

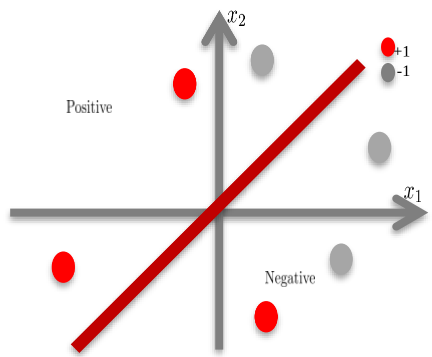
Is it the number of parameters?

Related but not really the same.

VC dimension for the linear classifiers we have learned so far.

$$f(\mathbf{x};\mathbf{w},b)=\begin{cases}+1 & if\ \mathbf{w}^T\mathbf{x}+b\geq 0\\ -1 & otherwise\end{cases}$$

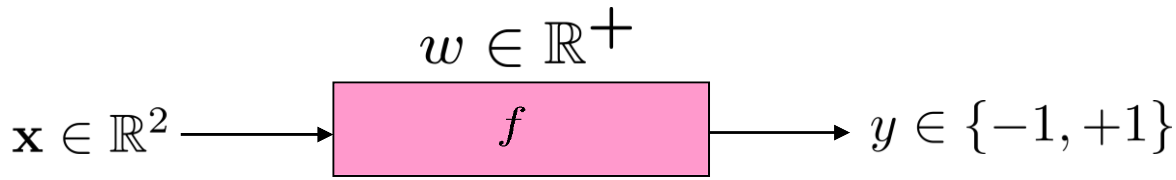
$$\mathbf{x},\mathbf{w}\in\mathbb{R}^r,\ b\in\mathbb{R}$$

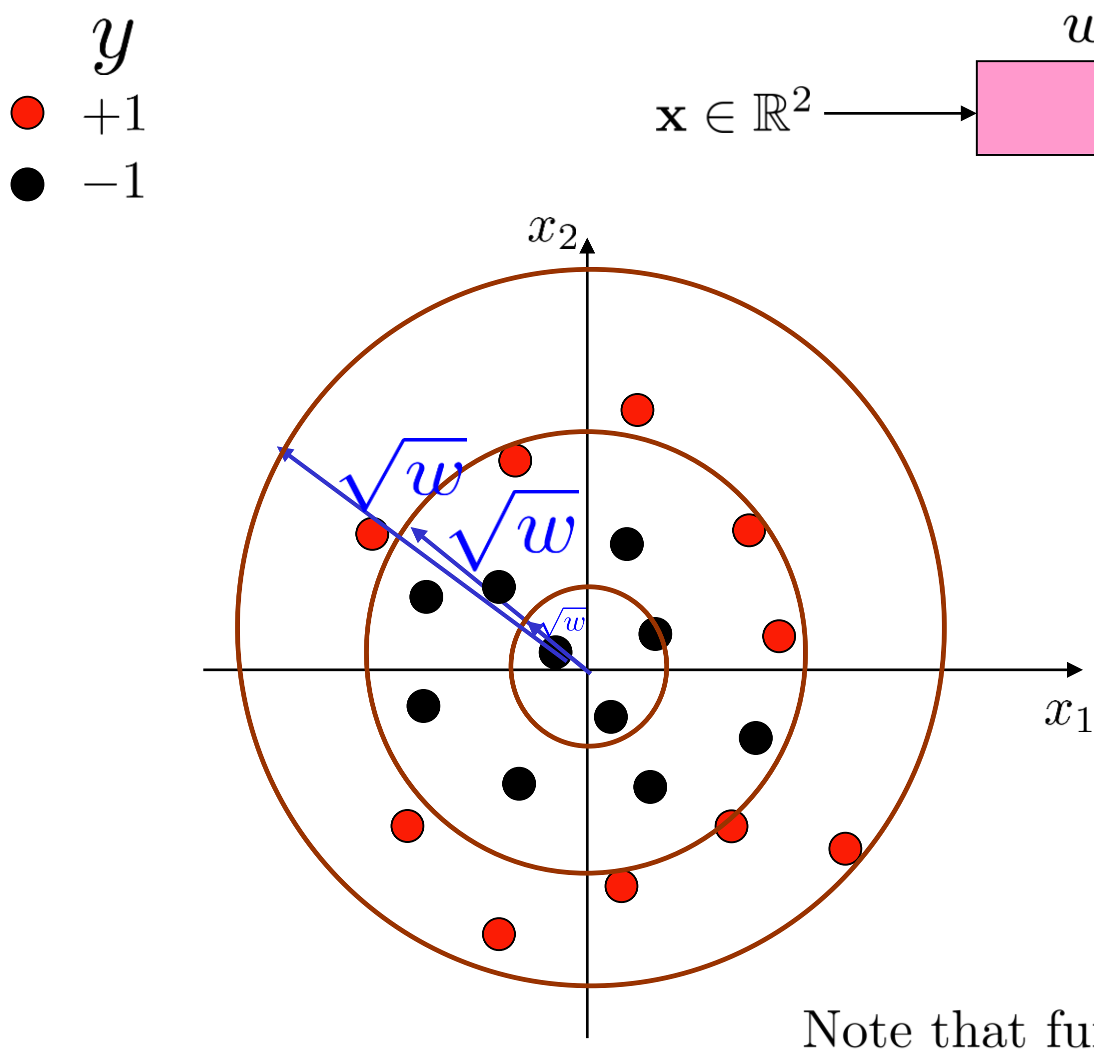


- Perceptron
- Logistic regression classifier
- Support vector machine (SVM)

Their VC dimension is $r+1$ in these linear classifier cases.

Example A



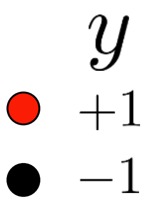


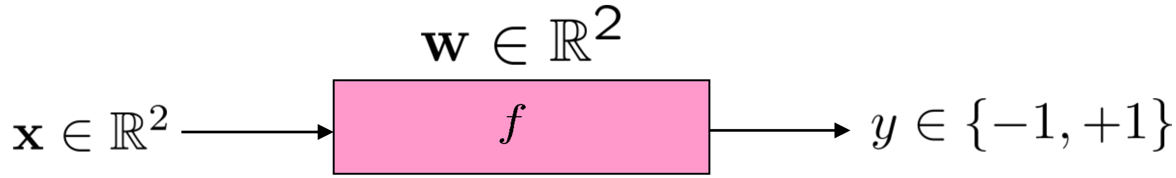
$$f(\mathbf{x}; w) = sign(\mathbf{x}^T\mathbf{x} - w)$$

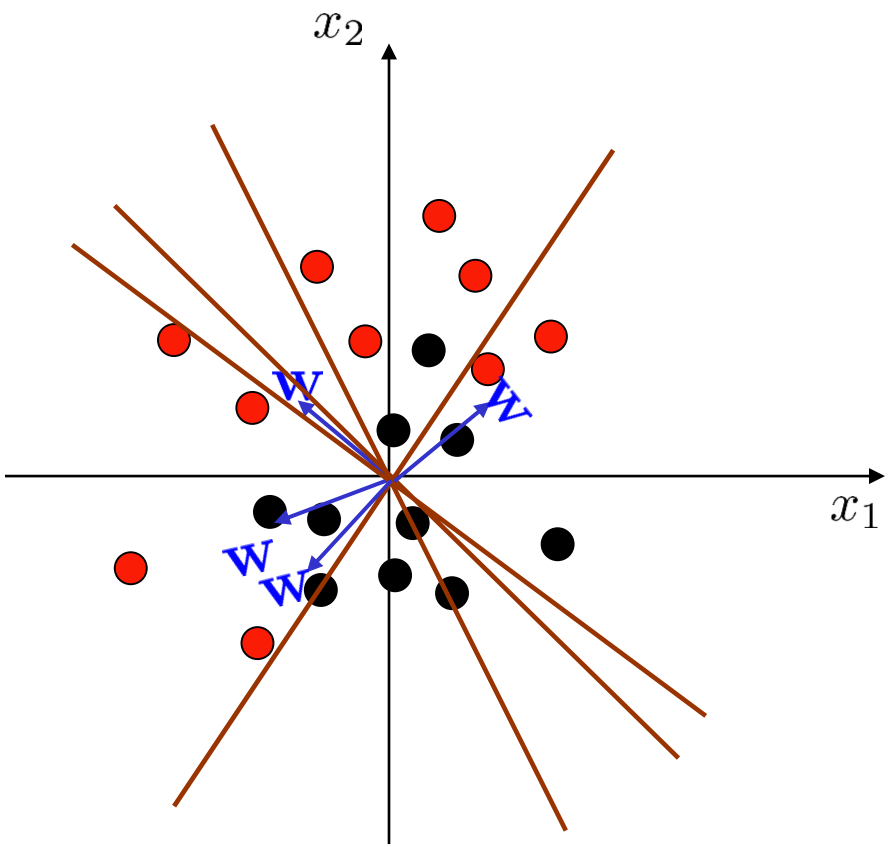
free parameter: $\exists w, w \in \mathbb{R}^+$.

Note that function $\mathbf{x}^T\mathbf{x} - w = 0$ defines a circle that has a radius $\sqrt{w}$.

Example B







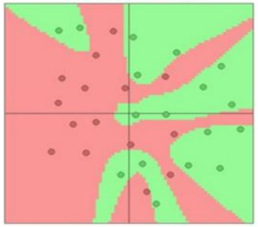
$$f(\mathbf{x}; w) = sign(\mathbf{w}^T\mathbf{x})$$

free parameter: $\exists \mathbf{w}, \mathbf{w} \in \mathbb{R}^2$.

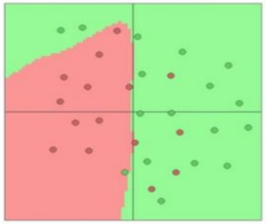
How do we characterize power?

- Different machines have different amounts of "power".
- Tradeoff between:

  More power: Can model more complex classifiers but might overfit.



  Less power: Not going to overfit, but restricted in what it can model.



- How do we characterize the amount of power?

Vapnik-Chervonenkis Dimension

$$e_{testing} \leq e_{training} + \sqrt{\frac{h(\log(2n/h+1)) - \log(\eta/4)}{n}}$$

- This gives us a way to estimate the error on future data based only on the training error and the VC-dimension (h) of **f**.
- This is a not so tight bound and it makes sense when n is big (e.g. > 10,000).
- In order to reduce the testing error, we need to reduce both the training error and the VC dimension of the classifier.
- h is the VC dimension but how do we compute h?

n: the number of training samples

$1 - \eta$: confidence level, can be ignored for the moment

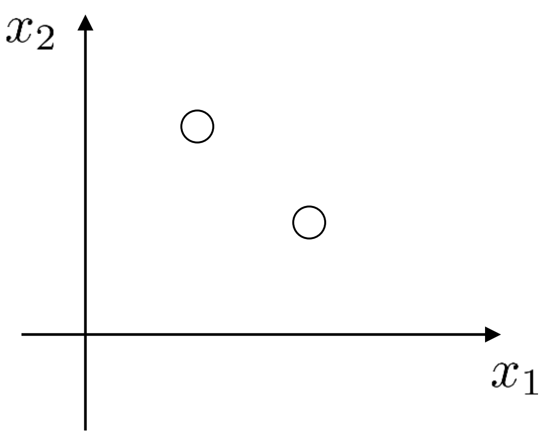
Shattering

Machine f can shatter a set of points $\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_r$ if and only if: for every possible $\{y_1, y_2, ..., y_r\}$ ($y_i \in \{-1, +1\}, i = 1..r$) form $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), , (\mathbf{x}_r, y_r)$, there exists some value of that gets zero training error.

There are 2^r such training sets to consider, each with a different combinations of +1s and -1s for the ys.

$$f(\mathbf{x}; \mathbf{w}) = sign(\mathbf{w}^T \mathbf{x}) \qquad \mathbf{x} \in \mathbb{R}^2$$

Free parameters: $\exists \mathbf{w}$ $\qquad \mathbf{w} \in \mathbb{R}^2$



Question: Can $f(\mathbf{x}; \mathbf{w})$ shatter the two points?

A. Yes
B. No
C. It depends

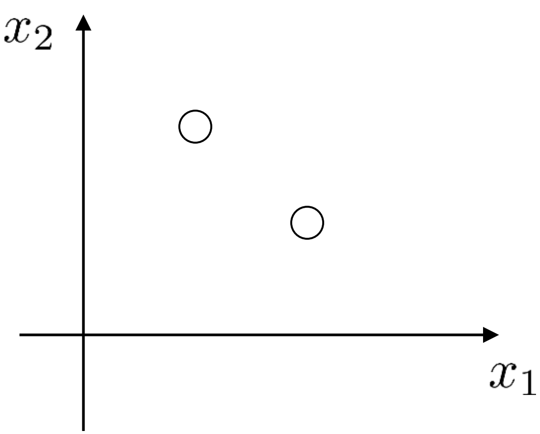
Shattering

Machine f can shatter a set of points $\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_r$ if and only if: for every possible $\{y_1, y_2, ..., y_r\}$ $(y_i \in \{-1, +1\}, i = 1..r)$ form $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), , (\mathbf{x}_r, y_r)$, there exists some value of that gets zero training error.

There are 2^r such training sets to consider, each with a different combinations of +1s and -1s for the ys.

$$f(\mathbf{x}; \mathbf{w}) = sign(\mathbf{w}^T \mathbf{x}) \qquad \mathbf{x} \in \mathbb{R}^2$$

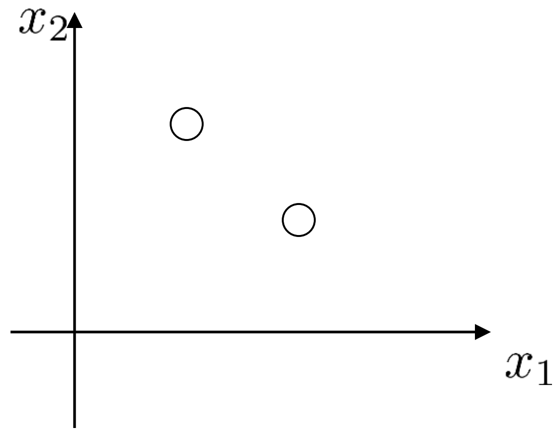
Free parameters: $\exists \mathbf{w}$ $\qquad \mathbf{w} \in \mathbb{R}^2$



Question: Can $f(\mathbf{x}; \mathbf{w})$ shatter the two points?

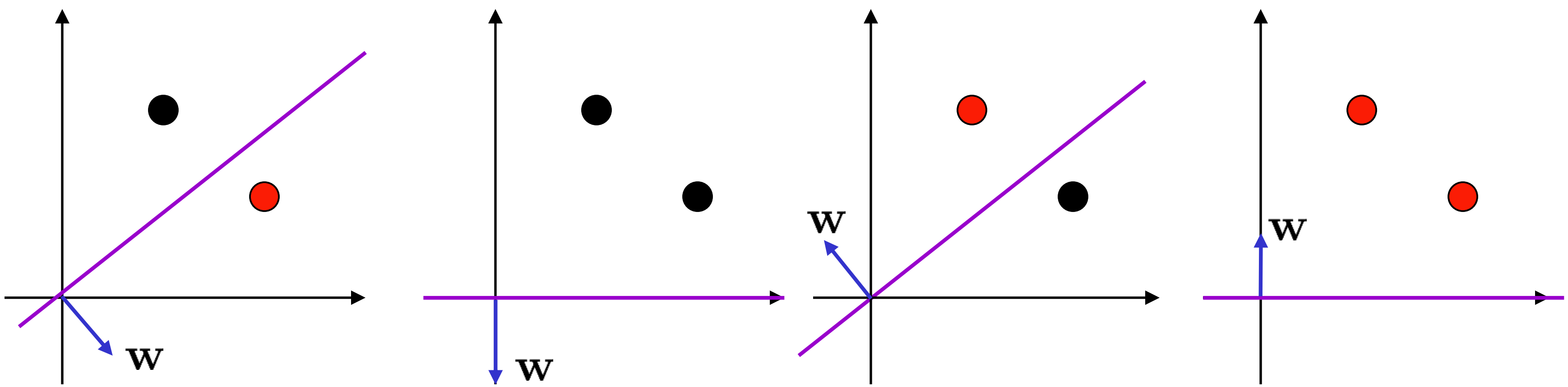
A. Yes
B. No
C. It depends

Shattering



$$f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x}) \quad \mathbf{x} \in \mathbb{R}^2$$

Free parameters: $\exists \mathbf{w} \qquad \mathbf{w} \in \mathbb{R}^2$



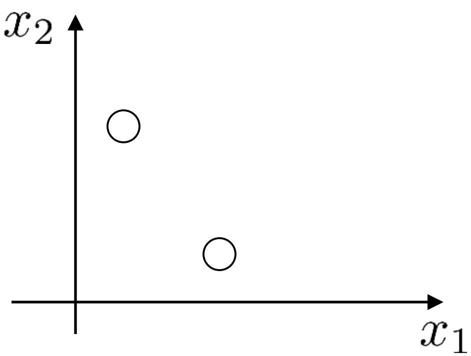
$\mathbf{w} = (1, -1)$ $\qquad$ $\mathbf{w} = (0, -1)$ $\qquad$ $\mathbf{w} = (-1, 1)$ $\qquad$ $\mathbf{w} = (0, 1)$

Shattering

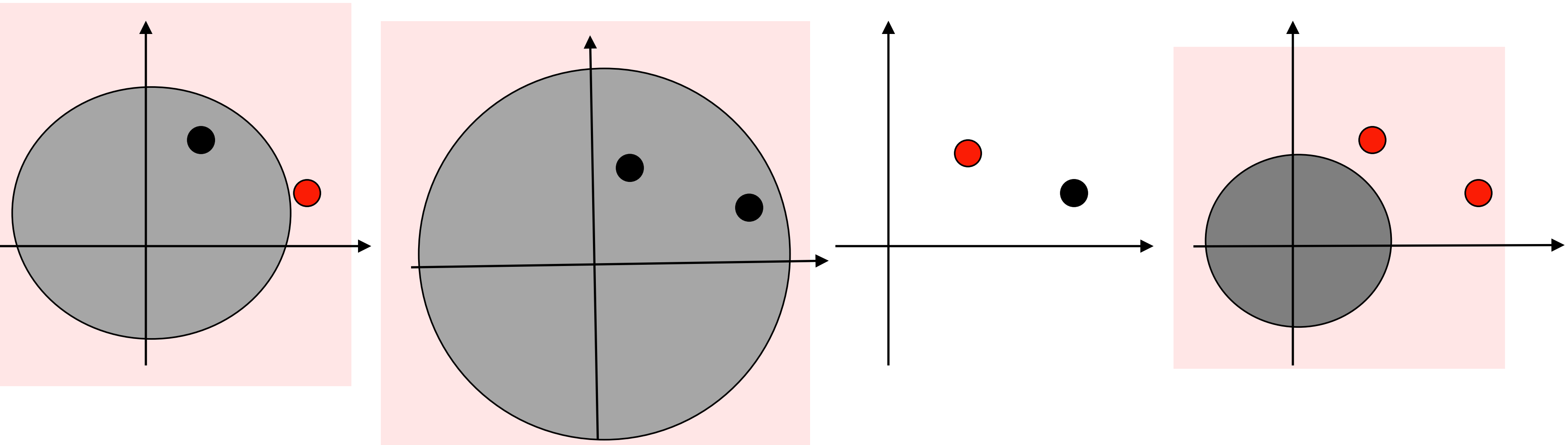
$f(\mathbf{x}; w) = sign(\mathbf{x}^T\mathbf{x} - w)$ $\quad \mathbf{x} \in \mathbb{R}^2$ Free parameters: $\exists w \in \mathbb{R}$

Note that function $\mathbf{x}^T\mathbf{x} - w = 0$ defines a circle that has a radius $\sqrt{w}$.

Question: Can the $f(\mathbf{x}; w)$ shatter the two points?



A. Yes
B. No
C. It depends



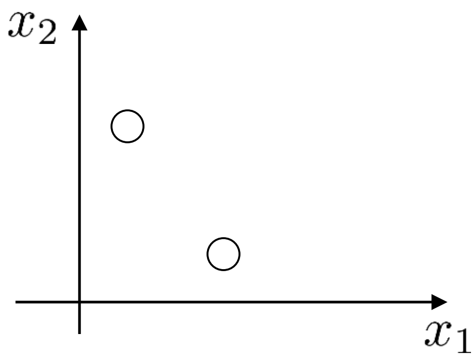
yes yes no yes

Shattering

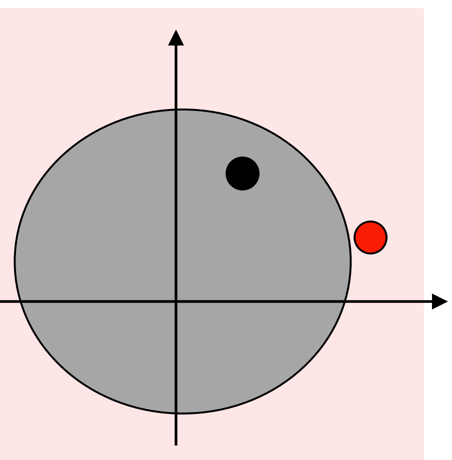
$f(\mathbf{x}; w) = sign(\mathbf{x}^T\mathbf{x} - w)$ Free parameters: $\exists w \in \mathbb{R}$

Note that function $\mathbf{x}^T\mathbf{x} - w = 0$ defines a circle that has a radius $\sqrt{w}$.

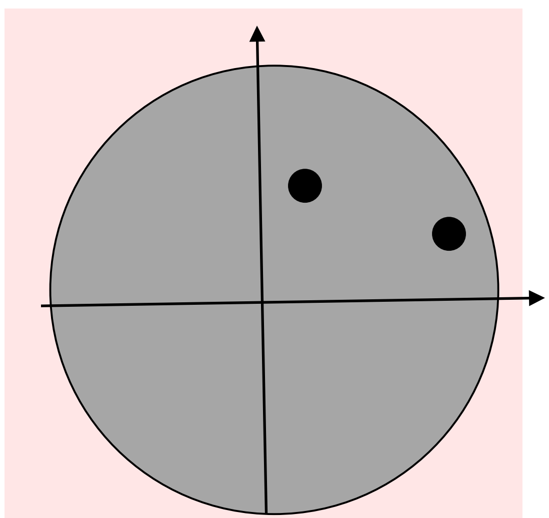
Question: Can the $f(\mathbf{x}; w)$ shatter the two points?



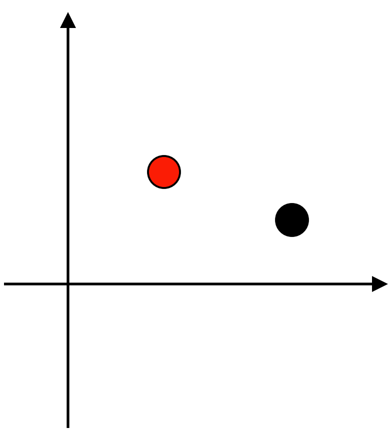
A. Yes
B. No
C. It depends



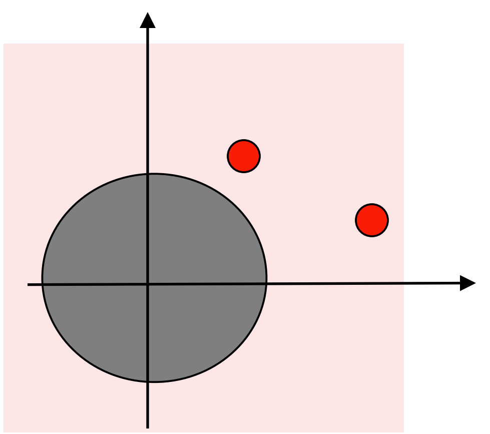
yes



yes

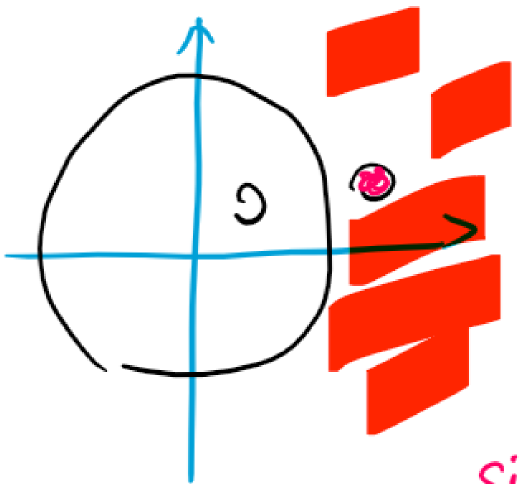


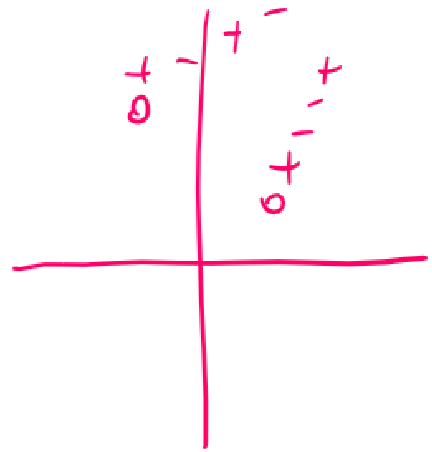
no



yes

$w^T x = x_0 - th > 0$

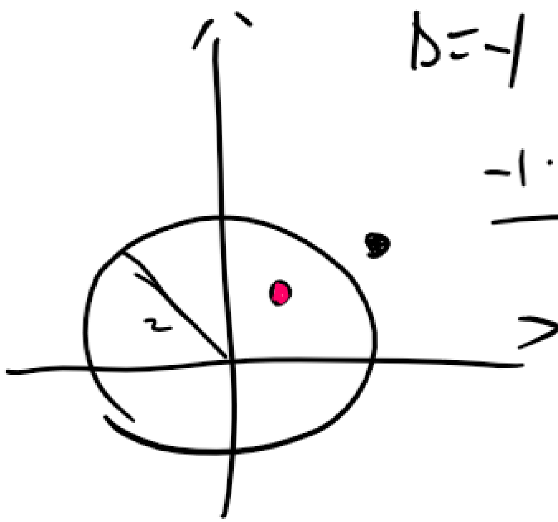




sign(x·x−w) can not shatter two points.

$f(x; w, b) = b \cdot \text{sign}(x \cdot x - w)$?

$b = -1$



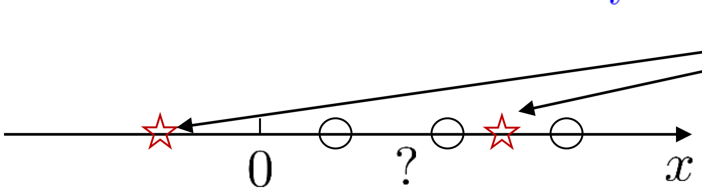
$-1 \cdot \text{sign}(x \cdot x - 4)$.

$- \cdot \text{sing}(1 - 4) = +$

Understanding shattering

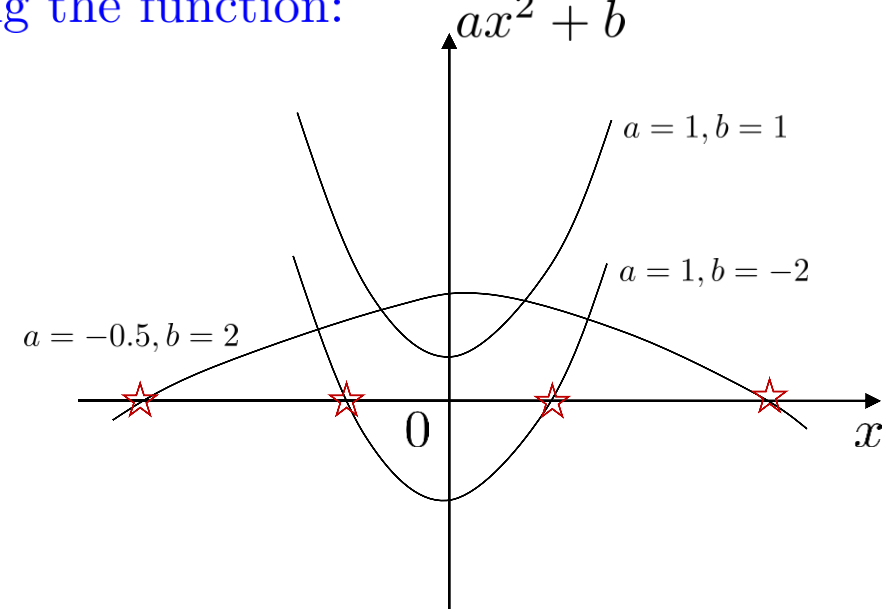
Example: What is the VC-dimension for $f(x;a,b) = sign(ax^2+b)$, $x,a,b \in \mathbb{R}$?

Understanding the problem: Decision boundary consists of a set of points on the axis.



e.g. for $\forall x$ such that $ax^2+b=0$.

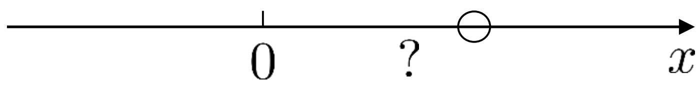
Understanding the function:

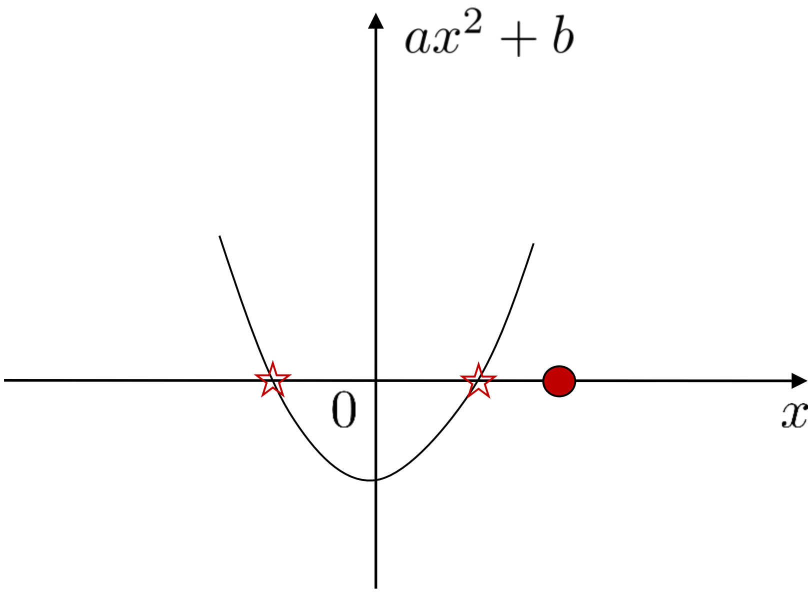


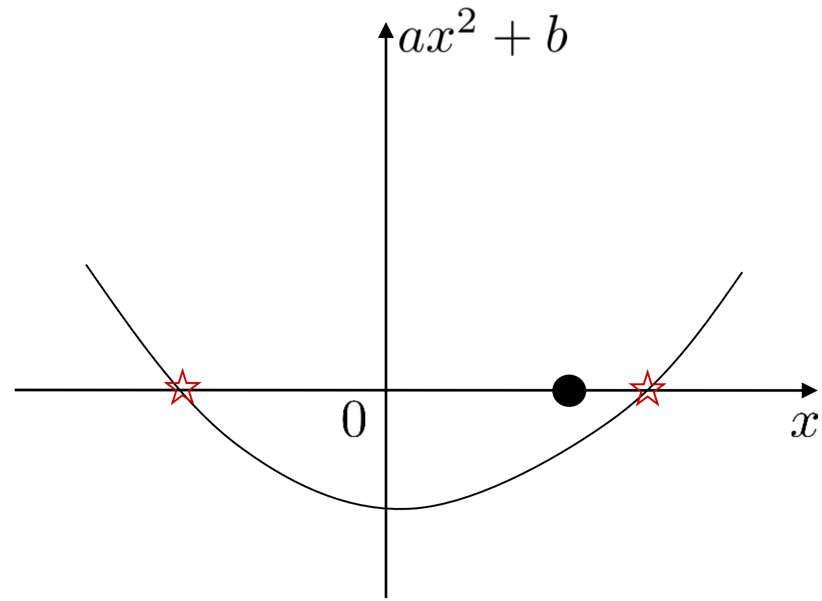
Understanding shattering

Example: What is the VC-dimension for $f(x; a, b) = sign(ax^2 + b)$, $x, a, b \in \mathbb{R}$?

One point:



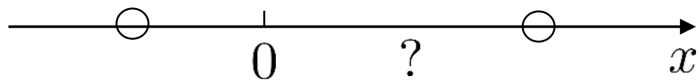


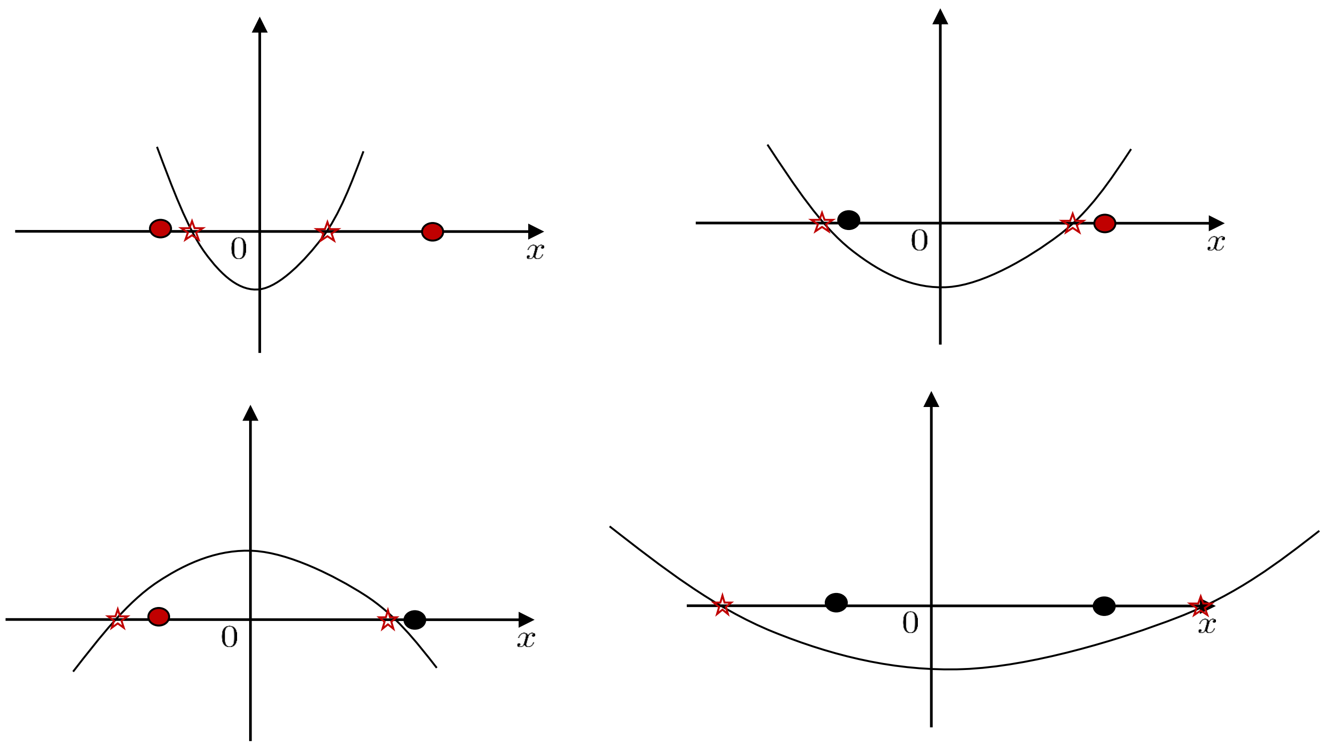


Understanding shattering

Example: What is the VC-dimension for $f(x; a, b) = sign(ax^2 + b)$, $x, a, b \in \mathbb{R}$?

Two points:





Intuition about classification power

$$e_{testing}^{(f)} = e_{training}^{(f)} + e_{gen}(f)$$

- Typically, more powerful a classifier f is, the smaller the training error it can achieve.

$$e_{training}^{(f)} \rightarrow 0$$

- However, more powerful a classifier f is, the larger the generalization error it incurs.

$$e_{gen}(f) \rightarrow 0.5$$

- The power of a classifier is dependent on the type of classifier (e.g. perceptron, decision tree, nearest neighborhood, etc.) and how many parameters are being learned.

- The power of a classifier doesn't depend on the exact optimal parameters learned after training on a specific task.

Intuition about shattering

- We want to come up a way to characterize the classification power of a given type of classifier that should be agnostic across ALL types of classifiers (disqualifying counting the number of parameters since they have different interpretations for different classifier types).

- Using the concept of shattering allows us to find out the capability of a classifier, given a number of non-overlapping points, by successfully classifying them under all possible labeling configurations.

- If you are checking on n points, then there are 2^n possibilities to verify. Failing on any one of the situations will deem the classifier incapable of shattering n points.

- This is like a bank stress test.

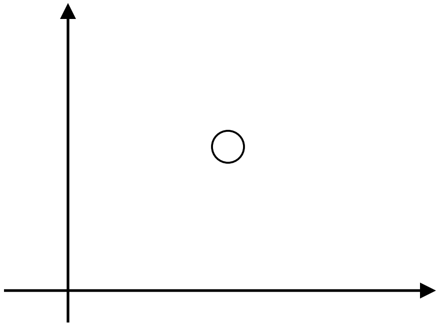
Definition of VC dimension

Given machine f, the VC-dimension h is: The maximum number of points that can (exists any situation) be arranged so that f can shatter them.

Example: what is the VC-dimension for $f(\mathbf{x}; w) = sign(\mathbf{x}^T\mathbf{x} - w)$? $\quad \mathbf{x} \in \mathbb{R}^2$

Free parameters: $\exists w \in \mathbb{R}$

Answer: 1



Note that function $\mathbf{x}^T\mathbf{x} - w = 0$ defines a circle that has a radius $\sqrt{w}$.

VC-dimension of a linear model

For a linear classifier:

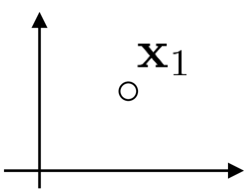
$$f(\mathbf{x}; \mathbf{w}, b) = sign(\mathbf{w}^T \mathbf{x} + b),$$

where $\mathbf{x} \in \mathbb{R}^2$ with free parameters: $\mathbf{w} \in \mathbb{R}^2, b \in \mathbb{R}$,

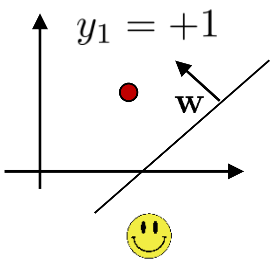
it's VC-dimension is 3.

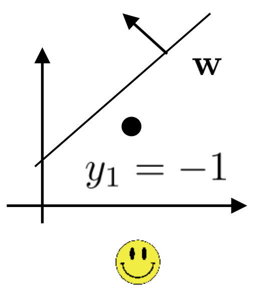
Proof:

1. For 1 point:



yes





VC-dimension of a linear model

For a linear classifier:

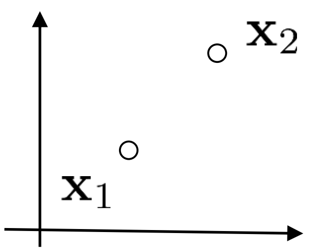
$$f(\mathbf{x}; \mathbf{w}, b) = sign(\mathbf{w}^T\mathbf{x} + b),$$

where $\mathbf{x} \in \mathbb{R}^2$ with free parameters: $\mathbf{w} \in \mathbb{R}^2, b \in \mathbb{R}$,

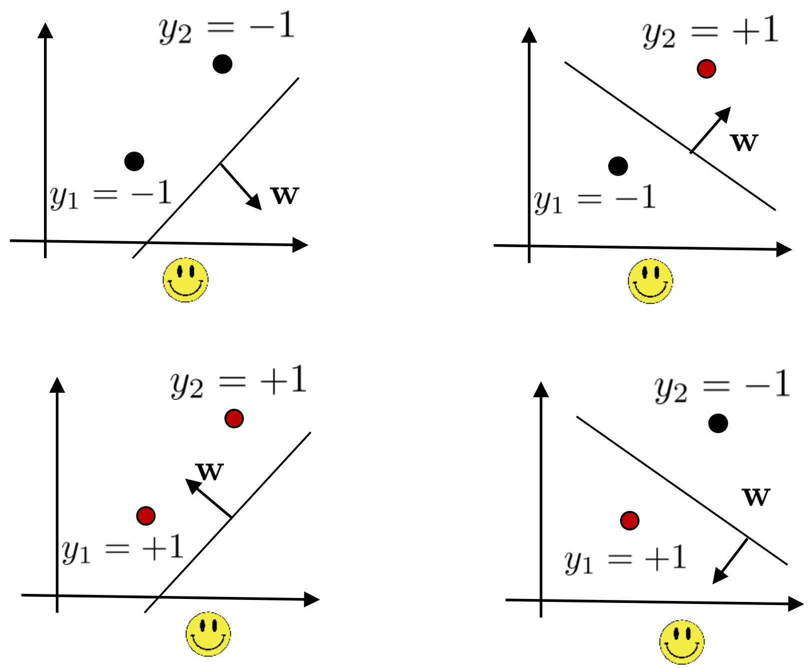
it's VC-dimension is 3.

Proof:

2. For 2 points:



yes



VC-dimension of a linear model

For a linear classifier:

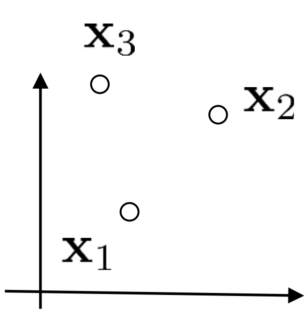
$$f(\mathbf{x}; \mathbf{w}, b) = sign(\mathbf{w}^T \mathbf{x} + b),$$

where $\mathbf{x} \in \mathbb{R}^2$ with free parameters: $\mathbf{w} \in \mathbb{R}^2, b \in \mathbb{R}$,

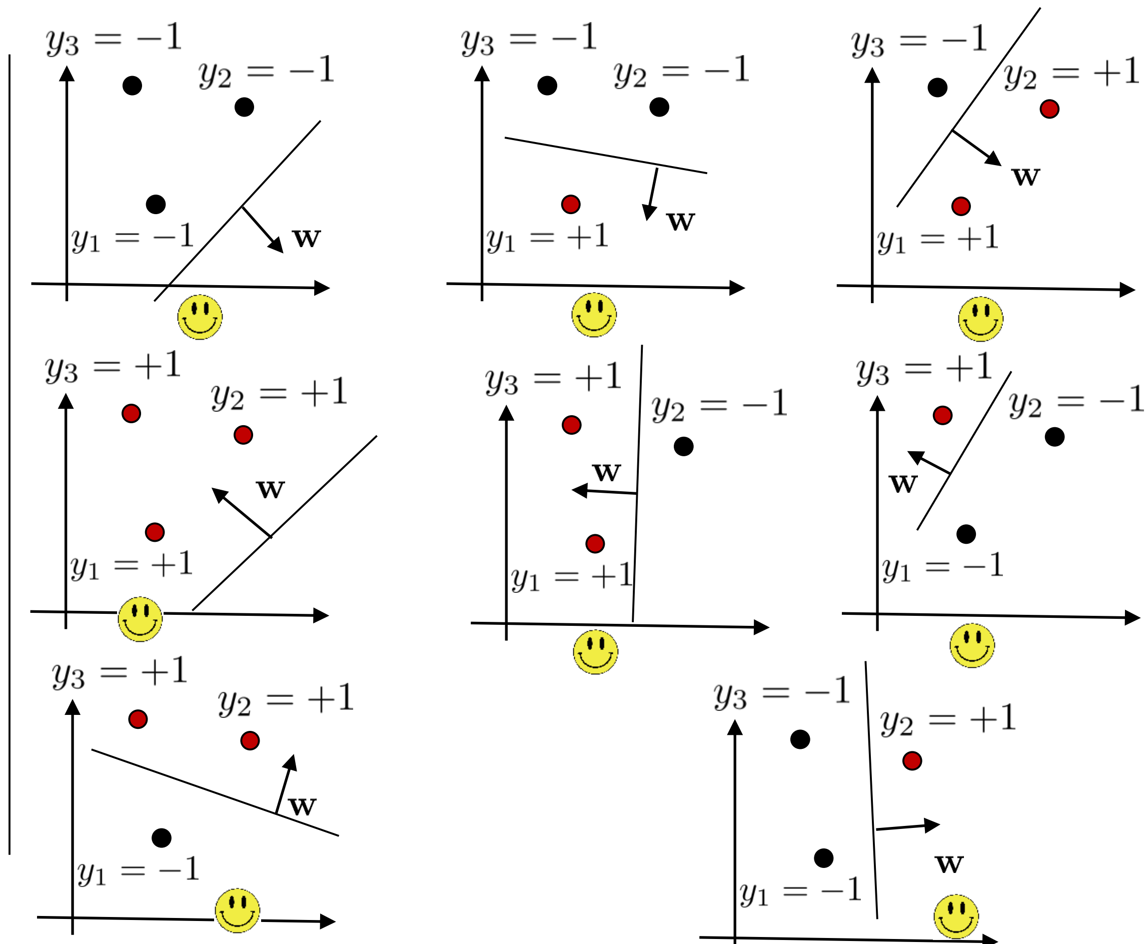
it's VC-dimension is 3.

Proof:

3. For 3 points:



yes



VC-dimension of a linear model

For a linear classifier:

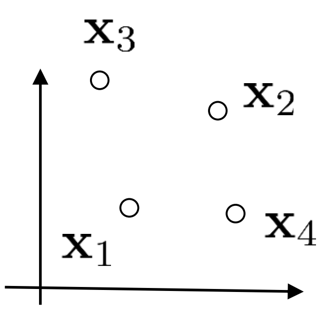
$$f(\mathbf{x}; \mathbf{w}, b) = sign(\mathbf{w}^T \mathbf{x} + b),$$

where $\mathbf{x} \in \mathbb{R}^2$ with free parameters: $\mathbf{w} \in \mathbb{R}^2, b \in \mathbb{R}$,

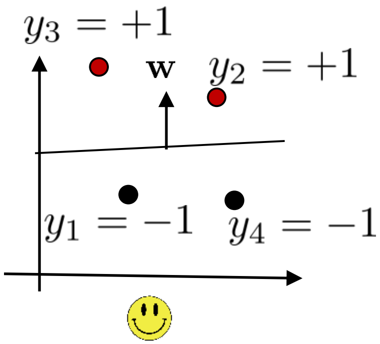
it's VC-dimension is 3.

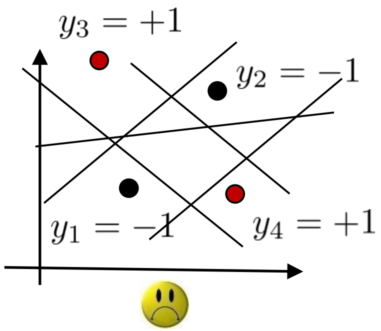
Proof:

4. For 4 points:



no





VC-dimension

Theory: The VC dimension (h) of the set of oriented hyperplanes in $\mathbb{R}^r$ is $r+1$, since we can always choose r+1 points, and then choose one of the points as origin, such that the position vectors of the remaining r points are linearly independent, but can never choose r+2 such points.

For a linear classifier:

$$f(\mathbf{x}; \mathbf{w}, b) = sign(\mathbf{w}^T\mathbf{x} + b),$$

$$\mathbf{x}, \mathbf{w} \in \mathbb{R}^r, \; b \in \mathbb{R}$$

Total number of parameters: $r+1$.

- VC dimension (h) reports the maximum number of points a classifier f can shatter.
- It is done by checking the number of shattering sequentially from 1,2,3,… until it fails.

VC Dimension

- The concept and theory of VC dimension, named after Vapnik and Chervonenkis, defines the maximum capability of a classifier f.

- VC dimension (h) reports the maximum number of points a classifier f can shatter.

- It is done by checking the number of shattering sequentially from 1,2,3,… until it fails.

- When checking on number n, you only need to find an existence of n non-overlapping points (no need to shatter all possible n points).

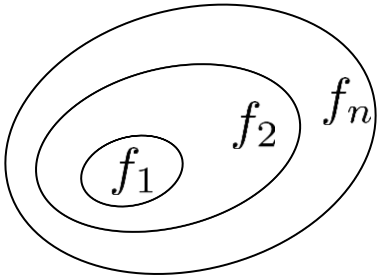
- However, once the n points are given, you need to make sure ALL possible labeling configurations for these n points can be well classified. Otherwise, it's a failure.

Structural Risk Minimization

Let: $\phi(f)$=the set of functions representable by f

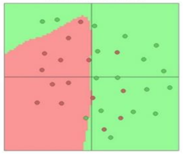
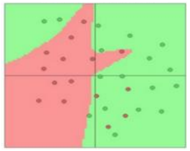
Suppose: $\phi(f_1) \subseteq \phi(f_2) \subseteq \cdots \phi(f_n)$

Then: $h(f_1) \leq h(f_2) \leq \cdots h(f_n)$



We are trying to decide which machine to use.

We train each machine and make a table: $e_{testing} \leq e_{training} + \sqrt{\frac{h(\log(2n/h+1) - \log(\eta/4)}{n}}$

	i	f_i	$e_{training}$	$\sqrt{\frac{h(\log(2n/h+1) - \log(\eta/4)}{n}}$ generalization	upper bound $e_{testing}$	choice
A	1	f_1				
B	2	f_2				
C	3	f_3				 
D	4	f_4				
E	5	f_5