CS182: Introduction to Machine Learning – Support Vector Machines (SVMs)

Yujiao Shi
SIST, ShanghaiTech
Spring, 2025

Slides Courtesy of Maria-Florina Balcan

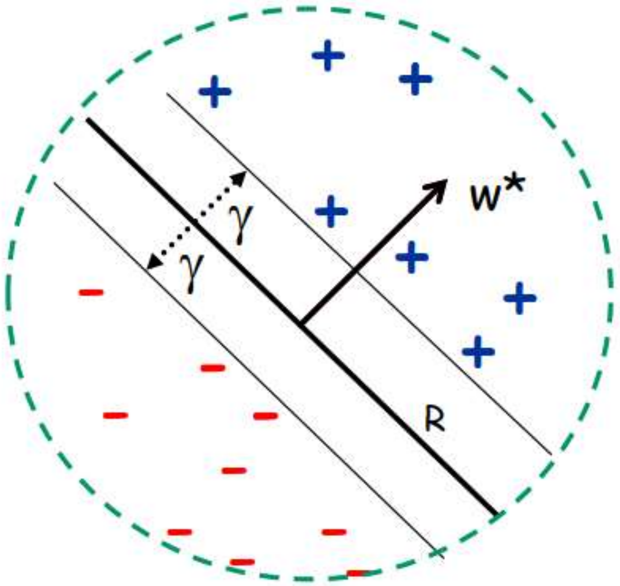
Please let me know your feedback on this course

https://wj.qq.com/s2/18474381/26b4/

Perceptron: Mistake Bound

Theorem: If data linearly separable by margin γ and points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

- No matter how long the sequence is how high dimension n is!



Margin: the amount of wiggle-room available for a solution.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes; algo is invariant to scaling.)

Perceptron Algorithm: Analysis

Theorem: If data has margin γ and all points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

Update rule:

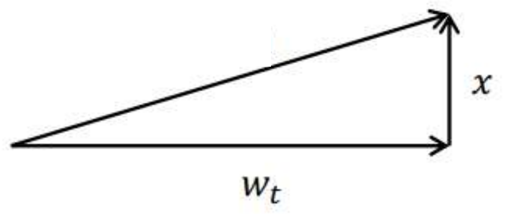
- Mistake on positive: $w_{t+1} \leftarrow w_t + x$
- Mistake on negative: $w_{t+1} \leftarrow w_t - x$

Proof:

Idea: analyze $w_t \cdot w^*$ and $\|w_t\|$, where w^* is the max-margin sep, $\|w^*\| = 1$.

Claim 1: $w_{t+1} \cdot w^* \geq w_t \cdot w^* + \gamma$.

Claim 2: $\|w_{t+1}\|^2 \leq \|w_t\|^2 + R^2$.

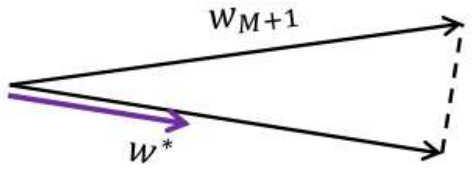


After M mistakes:

$w_{M+1} \cdot w^* \geq \gamma M$ (by Claim 1)

$\|w_{M+1}\| \leq R\sqrt{M}$ (by Claim 2)

$w_{M+1} \cdot w^* \leq \|w_{M+1}\|$ (since w^* is unit length)



So, $\gamma M \leq R\sqrt{M}$, so $M \leq \left(\frac{R}{\gamma}\right)^2$.

Perceptron: Mistake Bound

Theorem: If data linearly separable by margin γ and points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

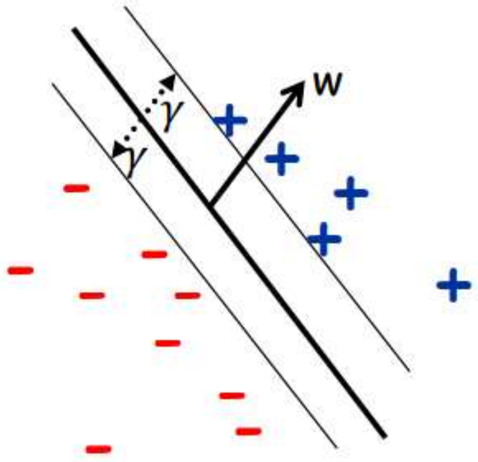
Implies that large margin classifiers have smaller complexity!

Margin Important Theme in ML

Both sample complexity and algorithmic implications.

Sample/Mistake Bound complexity:

- If large margin, # mistakes Peceptron makes is small (independent on the dim of the space)!
- If large margin γ and if alg. produces a large margin classifier, then amount of data needed depends only on R/γ [Bartlett & Shawe-Taylor '99].
    - Suggests searching for a large margin classifier…



Algorithmic Implications:

- Perceptron, Kernels, SVMs…

So far, talked about margins in the context of (nearly) linearly separable datasets

What if Not Linearly Separable?

上海科技大学 ShanghaiTech University

Problem: data not linearly separable in the most natural feature representation.

Example:

vs

No good linear separator in pixel representation.

Solutions:

- "Learn a more complex class of functions"
  - (e.g., decision trees, neural networks, boosting).
- "Use a Kernel" (a neat solution that attracted a lot of attention)
- "Use a Deep Network"
- "Combine Kernels and Deep Networks"

Overview of Kernel Methods

上海科技大学 ShanghaiTech University

What is a Kernel?

A kernel K is a legal def of dot-product: i.e. there exists an implicit mapping Φ s.t. K(, ) =Φ()$\cdot$ Φ()

E.g., $K(x,y) = (x \cdot y + 1)^d$

ϕ: (n-dimensional space) $\rightarrow$ n^d-dimensional space

Why Kernels matter?

- Many algorithms interact with data only via dot-products.
- So, if replace $x \cdot z$ with $K(x, z)$ they act implicitly as if data was in the higher-dimensional Φ-space.
- If data is linearly separable by large margin in the Φ-space, then good sample complexity.

  [Or other regularity properties for controlling the capacity.]

Kernels

Definition

$K(\cdot,\cdot)$ is a kernel if it can be viewed as a legal definition of inner product:

- $\exists\ \phi: X \to R^N$ s.t. $K(x, z) = \phi(x) \cdot \phi(z)$
  - Range of ϕ is called the Φ-space.
  - N can be very large.

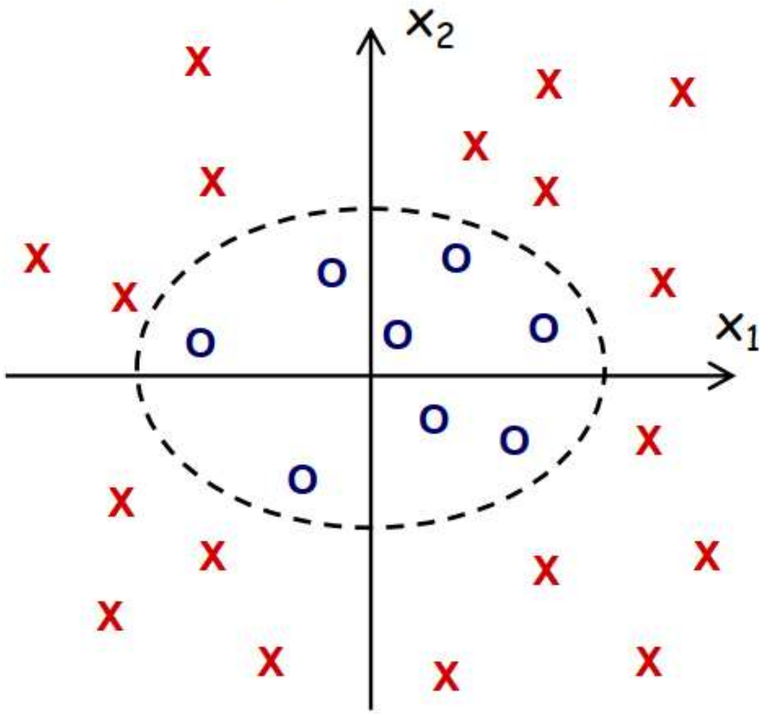
- But think of ϕ as implicit, not explicit!!!!

Example

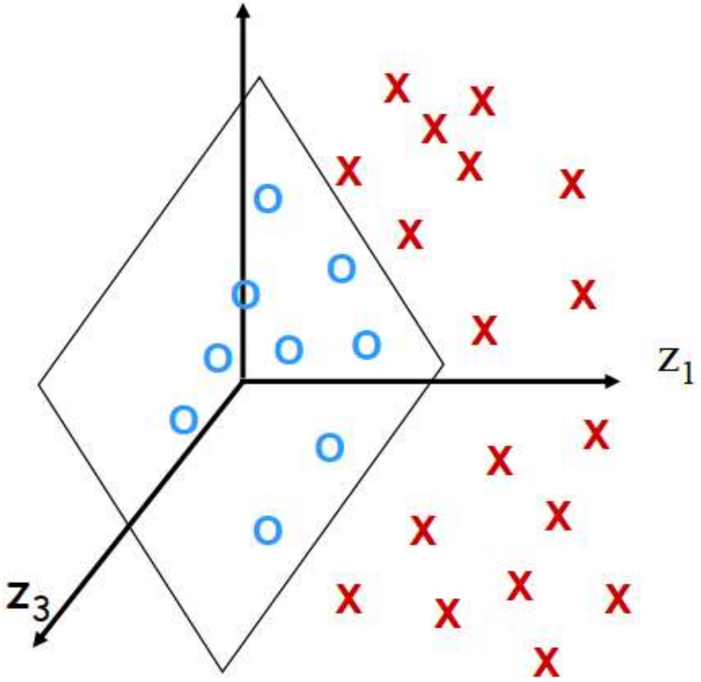
For n=2, d=2, the kernel $K(x, z) = (x \cdot z)^d$ corresponds to

$$(x_1, x_2) \rightarrow \Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

Original space



Φ-space

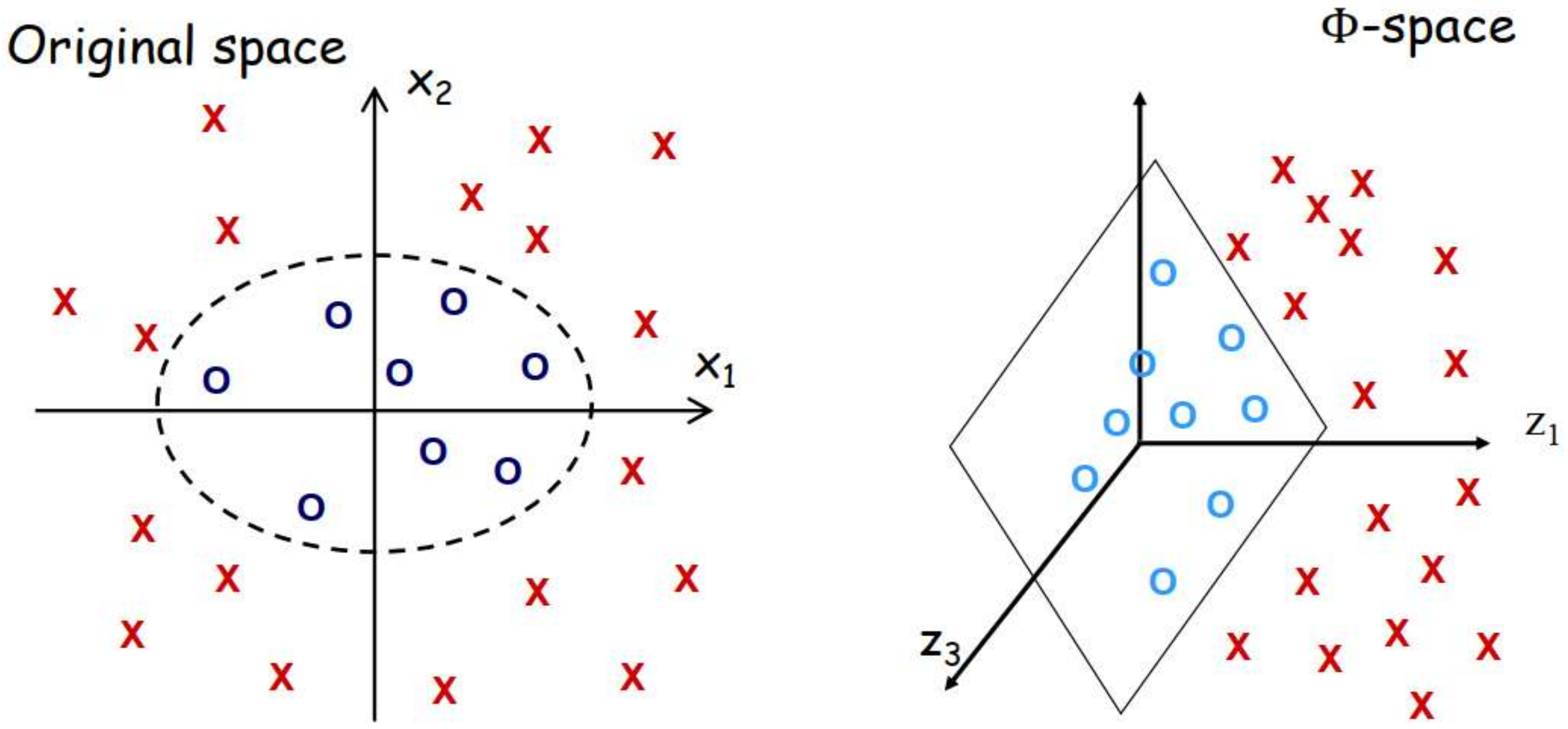


Example

$$\phi: R^2 \to R^3,\ (x_1, x_2) \to \Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(x) \cdot \phi(z) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2)$$

$$= (x_1z_1 + x_2z_2)^2 = (x \cdot z)^2 = K(x, z)$$



Example

上海科技大学 ShanghaiTech University

Note: feature space might not be unique.

$$\phi: R^2 \rightarrow R^3,\ (x_1, x_2) \rightarrow \Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(x)\cdot\phi(z) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)\cdot(z_1^2, z_2^2, \sqrt{2}z_1z_2)$$

$$= (x_1z_1 + x_2z_2)^2 = (x\cdot z)^2 = K(x,z)$$

$$\phi: R^2 \rightarrow R^4,\ (x_1, x_2) \rightarrow \Phi(x) = (x_1^2, x_2^2, x_1x_2, x_2x_1)$$

$$\phi(x)\cdot\phi(z) = (x_1^2, x_2^2, x_1x_2, x_2x_1)\cdot(z_1^2, z_2^2, z_1z_2, z_2z_1)$$

$$= (x\cdot z)^2 = K(x,z)$$

Avoid Explicitly Expanding the Features

上海科技大学
ShanghaiTech University

Feature space can grow really large and really quickly....

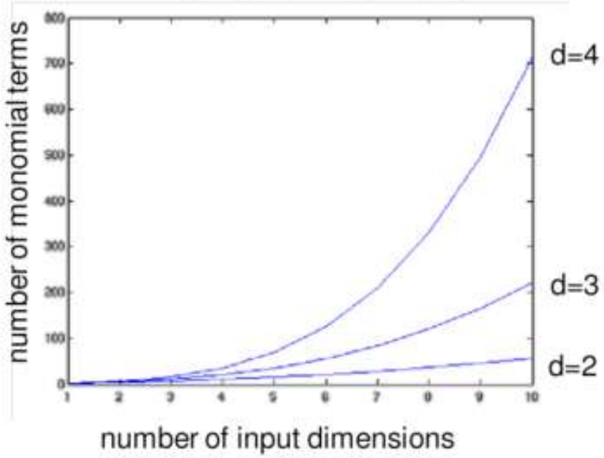
Crucial to think of ϕ as implicit, not explicit!!!!

Polynomial kernel degreee d, $k(x,z) = (x^\top z)^d = \phi(x) \cdot \phi(z)$

- $x_1^d, x_1 x_2 \dots x_d, x_1^2 x_2 \dots x_{d-1}$
- Total number of such feature is

$$\binom{d+n-1}{d} = \frac{(d+n-1)!}{d!\,(n-1)!}$$

- $d = 6, n = 100$, there are 1.6 billion terms



$O(n)$ computation!

$$k(x,z) = (x^\top z)^d = \phi(x) \cdot \phi(z)$$

Kernelizing a Learning Algorithm

上海科技大学 ShanghaiTech University

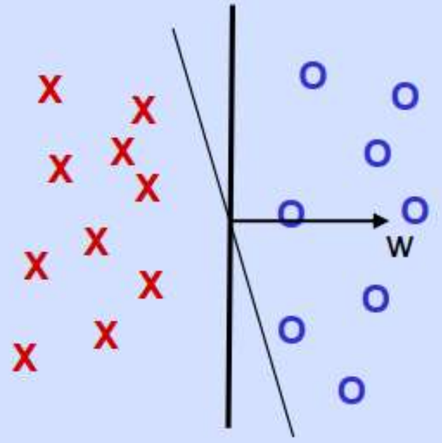
- **If all computations involving instances are in terms of inner products then:**
  - Conceptually, work in a very high diml space and the alg's performance depends only on linear separability in that extended space.
  - Computationally, only need to modify the algo by replacing each $x \cdot z$ with a $K(x, z)$.

- **Examples of kernalizable algos:**
  - classification: Perceptron, SVM.
  - regression: linear, ridge regression.
  - clustering: k-means.

Kernelizing the Perceptron Algorithm

上海科技大学 ShanghaiTech University

- Set t=1, start with the all zero vector w_1.
- Given example x, predict + iff $w_t \cdot x \geq 0$
- On a mistake, update as follows:
  - Mistake on positive, $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, $w_{t+1} \leftarrow w_t - x$



Easy to kernelize since w_t is weighted sum of incorrectly classified examples $w_t = a_{i_1} x_{i_1} + \cdots + a_{i_k} x_{i_k}$

Replace $w_t \cdot x = a_{i_1} x_{i_1} \cdot x + \cdots + a_{i_k} x_{i_k} \cdot x$ with

$a_{i_1} K(x_{i_1}, x) + \cdots + a_{i_k} K(x_{i_k}, x)$

Note: need to store all the mistakes so far.

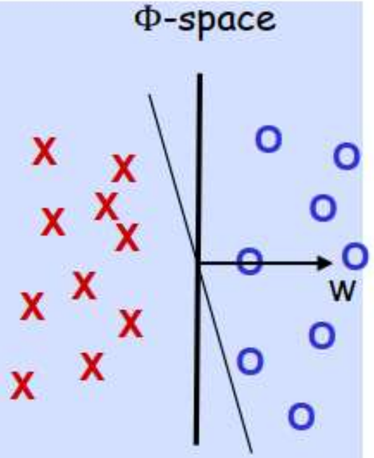
Kernelizing the Perceptron Algorithm

- Given x, predict + iff

$$\phi(x_{i_{t-1}}) \cdot \phi(x)$$

$$a_{i_1} K(x_{i_1}, x) + \cdots + a_{i_{t-1}} K(x_{i_{t-1}}, x) \geq 0$$

- On the t th mistake, update as follows:
  - Mistake on positive, set $a_{i_t} \leftarrow 1$; store x_{i_t}
  - Mistake on negative, $a_{i_t} \leftarrow -1$; store x_{i_t}



Perceptron $w_t = a_{i_1} x_{i_1} + \cdots + a_{i_k} x_{i_k}$

$$w_t \cdot x = a_{i_1} x_{i_1} \cdot x + \cdots + a_{i_k} x_{i_k} \cdot x \quad \rightarrow \quad a_{i_1} K(x_{i_1}, x) + \cdots + a_{i_k} K(x_{i_k}, x)$$

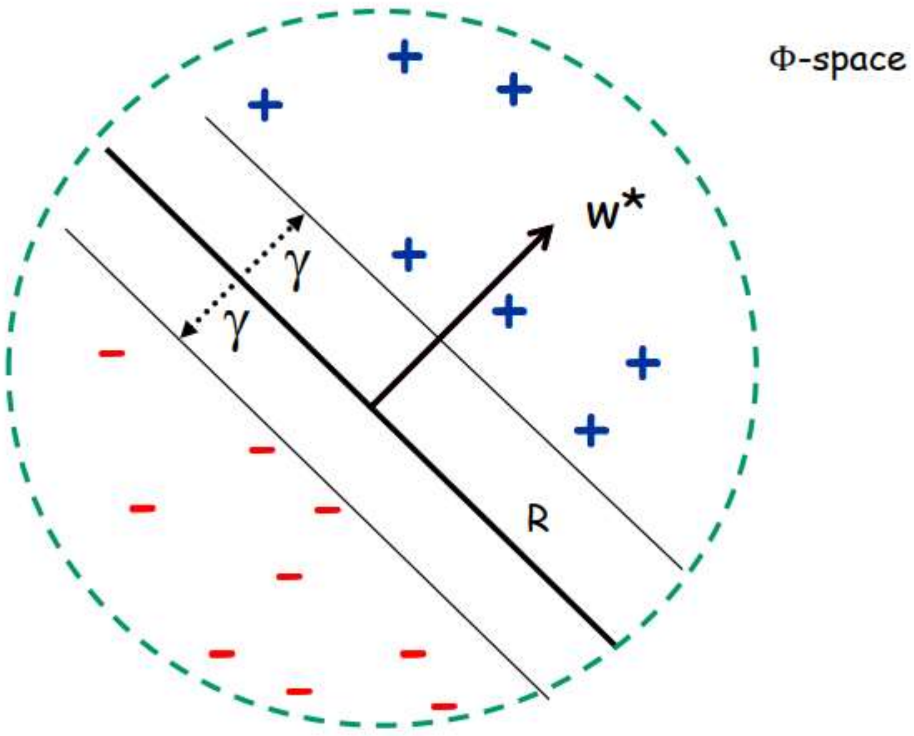
Exact same behavior/prediction rule as if mapped data in the ϕ-space and ran Perceptron there!

Do this implicitly, so computational savings!!!!!

Generalize Well if Good Margin

上海科技大学 ShanghaiTech University

- If data is linearly separable by margin in the ϕ-space, then small mistake bound.
- If margin γ in ϕ-space, then Perceptron makes $\left(\frac{R}{\gamma}\right)^2$ mistakes.



Kernels: More Examples

- Linear: $\mathrm{K}(\mathrm{x}, \mathrm{z}) = \mathrm{x} \cdot z$
- Polynomial: $\mathrm{K}(\mathrm{x}, z) = (\mathrm{x} \cdot z)^{\mathrm{d}}$ or $\mathrm{K}(\mathrm{x}, z) = (1 + \mathrm{x} \cdot z)^{\mathrm{d}}$
- Gaussian: $\mathrm{K}(\mathrm{x}, z) = \exp\left[-\frac{||x-z||^2}{2\,\sigma^2}\right]$
- Laplace Kernel: $\mathrm{K}(\mathrm{x}, z) = \exp\left[-\frac{||x-z||}{2\,\sigma^2}\right]$

- Kernel for non-vectorial data, e.g., measuring similarity between sequences.

Kernels: More Examples

Polynomial Kernel. The k-degree polynomial kernel is defined to be

$$K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k.$$

To see that this is indeed a kernel function, i.e. there exists mapping ψ for which $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$, denote $x_0 = x'_0 = 1$ and expand $K(\mathbf{x}, \mathbf{x}')$ as

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k \\ &= \left(\sum_{j=0}^{m} x_j x'_j \right) \cdot \left(\sum_{j=0}^{m} x_j x'_j \right) \cdot \cdots \cdot \left(\sum_{j=0}^{m} x_j x'_j \right) \\ &= \sum_{J \in [0:m]^k} \prod_{i=1}^{k} x_{J_i} x'_{J_i} \\ &= \sum_{J \in [0:m]^k} \prod_{i=1}^{k} x_{J_i} \cdot \prod_{i=1}^{k} x'_{J_i}. \end{aligned}$$

This is precisely the inner product $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ in the feature space if we define ψ to be

$$\psi(\mathbf{x}) = \left\{ \prod_{i=1}^{k} x_{J_i} \right\}_{J \in [0:m]^k}.$$

Note that here the complexity of implementing K is $O(m)$ while the dimension of the feature space is $(m+1)^k$.

Kernels: More Examples

上海科技大学 ShanghaiTech University

Gaussian Kernel. Let the original space $\mathcal{X} = \mathbb{R}$ and consider the mapping ψ given by

$$\psi(x) = \left\{ \frac{1}{\sqrt{m!}} e^{-\frac{x^2}{2}} x^m \right\}_{m=0}^{\infty}.$$

Then we have

$$\begin{aligned} \langle \psi(x), \psi(x') \rangle &= \sum_{m=0}^{\infty} \left(\frac{1}{\sqrt{m!}} e^{-\frac{x^2}{2}} x^m \right) \left(\frac{1}{\sqrt{m!}} e^{-\frac{(x')^2}{2}} (x')^m \right) \\ &= e^{-\frac{x^2 + (x')^2}{2}} \sum_{m=0}^{\infty} \left(\frac{(xx')^m}{m!} \right) \\ &= e^{-\frac{(x-x')^2}{2}}. \end{aligned}$$

Define the Gaussian kernel $K(x, x') = e^{-\frac{(x-x')^2}{2}}$. Obviously, evaluating the Gaussian kernel is very simple while in sharp contrast the feature space is of infinite dimension. Note that since $\psi(x)$ includes all the monomial terms, using the Gaussian kernel we can learn polynomial predictor of any degree over the original space.

Properties of Kernels

上海科技大学
ShanghaiTech University

Theorem (Mercer)

K is a kernel if and only if:

- K is symmetric
- For any set of training points $x_1, x_2, \dots, x_m$ and for any $a_1, a_2, \dots, a_m \in R$, we have:

$$\sum_{i,j} a_i a_j K(x_i, x_j) \geq 0$$

$$a^T K a \geq 0$$

I.e., $K = (K(x_i, x_j))_{i,j=1,\dots,n}$ is positive semi-definite.

Kernel Methods

- Offer great modularity. 
- No need to change the underlying learning algorithm to accommodate a particular choice of kernel function.
- Also, we can substitute a different algorithm while maintaining the same kernel.

Kernel, Closure Properties

Easily create new kernels using basic ones!

Fact: If $K_1(\cdot,\cdot)$ and $K_2(\cdot,\cdot)$ are kernels $c_1 \geq 0, c_2 \geq 0$,

then $K(x, z) = c_1 K_1(x, z) + c_2 K_2(x, z)$ is a kernel.

Key idea: concatenate the ϕ spaces.

$$\phi(x) = (\sqrt{c_1}\, \phi_1(x), \sqrt{c_2}\, \phi_2(x))$$

$$\phi(x) \cdot \phi(z) = c_1\, \underbrace{\phi_1(x) \cdot \phi_1(z)}_{K_1(x,z)} + c_2\, \underbrace{\phi_2(x) \cdot \phi_2(z)}_{K_2(x,z)}$$

Kernel, Closure Properties

Easily create new kernels using basic ones!

Fact: If $K_1(\cdot,\cdot)$ and $K_2(\cdot,\cdot)$ are kernels,

then $K(x,z) = K_1(x,z)K_2(x,z)$ is a kernel.

Key idea: $\phi(x) = \left(\phi_{1,i}(x)\ \phi_{2,j}(x)\right)_{i\in\{1,\ldots,n\},j\in\{1,\ldots,m\}}$

$$\phi(x)\cdot\phi(z) = \sum_{i,j} \phi_{1,i}(x)\ \phi_{2,j}(x)\phi_{1,i}(z)\ \phi_{2,j}(z)$$

$$= \sum_{i} \phi_{1,i}(x)\ \phi_{1,i}(z)\left(\sum_{j} \phi_{2,j}(x)\ \phi_{2,j}(z)\right)$$

$$= \sum_i \phi_{1,i}(x)\ \phi_{1,i}(z)K_2(x,z) = K_1(x,z)\ K_2(x,z)$$

Kernels, Discussion

上海科技大学 ShanghaiTech University

- If all computations involving instances are in terms of inner products then:
  - Conceptually, work in a very high diml space and the alg's performance depends only on linear separability in that extended space.
  - Computationally, only need to modify the algo by replacing each $x \cdot z$ with a $K(x, z)$.

- Lots of Machine Learning algorithms are kernalizable:
  - classification: Perceptron, SVM.
  - regression: linear regression.
  - clustering: k-means.

Kernels, Discussion

- If all computations involving instances are in terms of inner products then:
  - Conceptually, work in a very high diml space and the alg's performance depends only on linear separability in that extended space.
  - Computationally, only need to modify the algo by replacing each $x \cdot z$ with a $K(x, z)$.

How to choose a kernel:

- Kernels often encode domain knowledge (e.g., string kernels)
- Use Cross-Validation to choose the parameters, e.g., σ for Gaussian Kernel $K(x,z) = \exp\left[-\frac{||x-z||^2}{2\,\sigma^2}\right]$
- Learn a good kernel; e.g., [Lanckriet-Cristianini-Bartlett-El Ghaoui-Jordan'04]

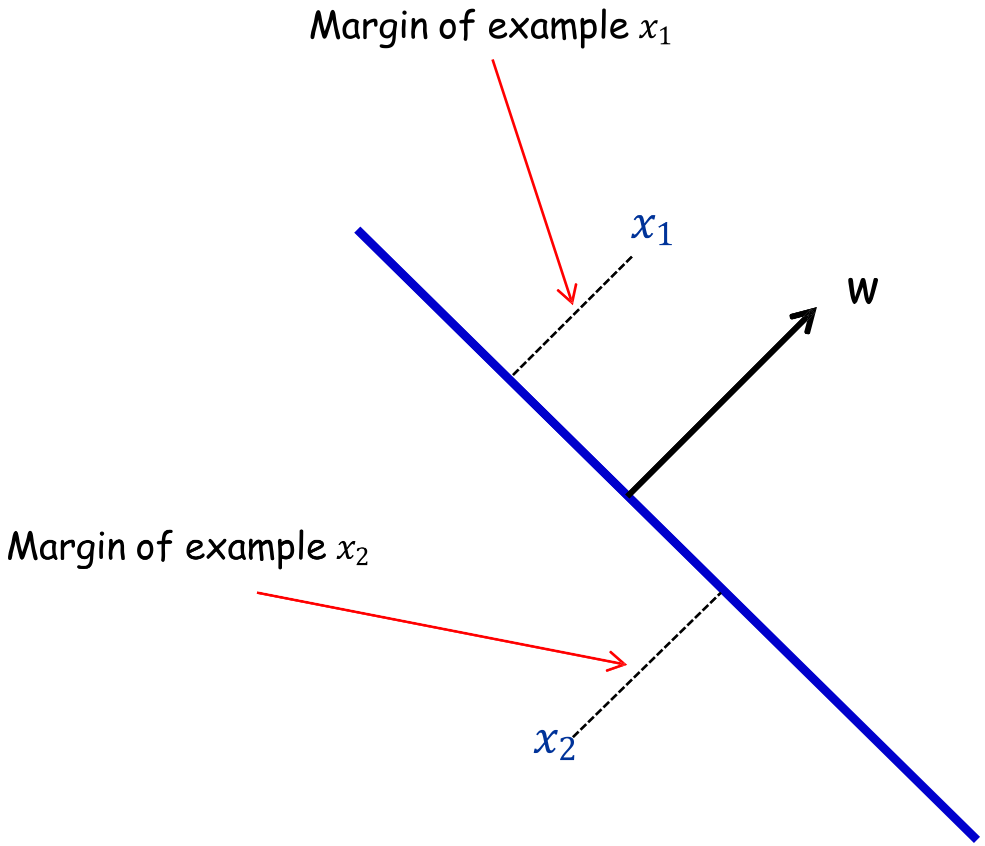
Support Vector Machines

- One of the most theoretically well motivated and practically most effective classification algorithms in machine learning.
- Directly motivated by Margins and Kernels!

Geometric Margin

上海科技大学 ShanghaiTech University

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$.



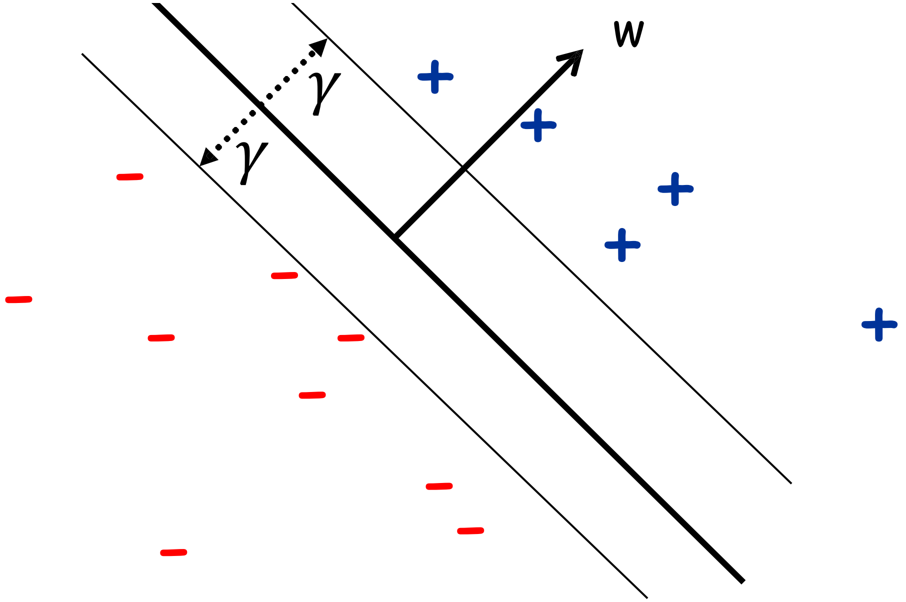
If $||w|| = 1$, margin of x w.r.t. w is $|x \cdot w|$.

Geometric Margin

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$.

Definition: The margin γ_w of a set of examples S wrt a linear separator w is the smallest margin over points $x \in S$.

Definition: The margin γ of a set of examples S is the maximum γ_w over all linear separators w.

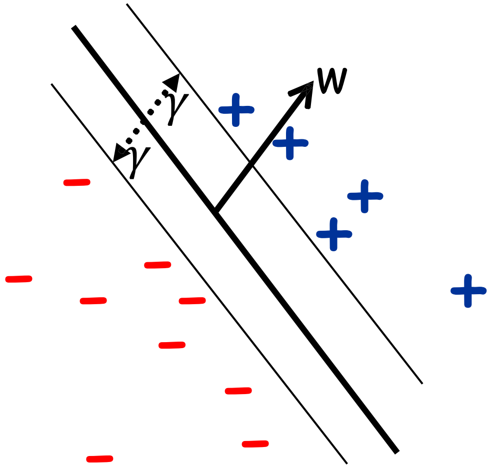


Margin Important Theme in ML

Both sample complexity and algorithmic implications.

Sample/Mistake Bound complexity:

- If large margin, # mistakes Peceptron makes is small (independent on the dim of the space)!
- If large margin γ and if alg. produces a large margin classifier, then amount of data needed depends only on R/γ [Bartlett & Shawe-Taylor '99].



Algorithmic Implications

Suggests searching for a large margin classifier… SVMs

Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

Directly optimize for the maximum margin separator: SVMs

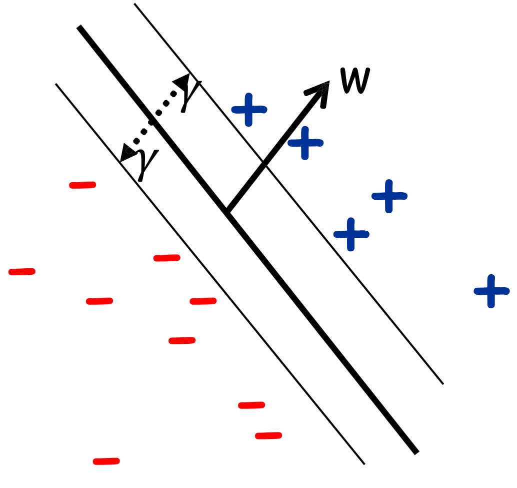
First, assume we know a lower bound on the margin γ

Input: γ, S={(x_1, y_1), …,(x_m, y_m)};

Find: some w where:

- $||w||^2 = 1$
- For all i, $y_i w \cdot x_i \geq \gamma$

Output: w, a separator of margin γ over S



Realizable case, where the data is linearly separable by margin γ

Support Vector Machines (SVMs)

Directly optimize for the maximum margin separator: SVMs

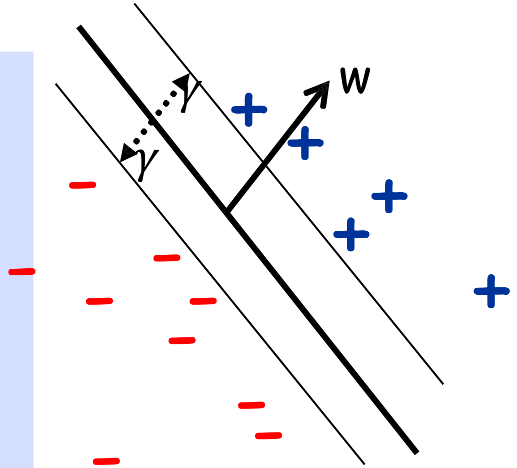
E.g., search for the best possible γ

<u>Input</u>: S={(x_1, y_1), …,$(\mathrm{x}_\mathrm{m}, y_\mathrm{m})$};

<u>Find</u>: some w and maximum γ where:

- $||\mathrm{w}||^2 = 1$
- For all i, $y_i w \cdot x_i \geq \gamma$

<u>Output</u>: maximum margin separator over S



Support Vector Machines (SVMs)

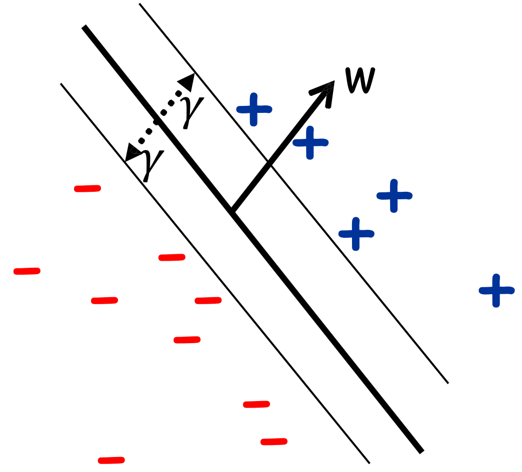
上海科技大学 ShanghaiTech University

Directly optimize for the maximum margin separator: SVMs

Input: S={(x_1, y_1), …,(x_m, y_m)};

Maximize γ under the constraint:

- $||w||^2 = 1$
- For all i, $y_i w \cdot x_i \geq \gamma$



Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

Directly optimize for the maximum margin separator: SVMs

Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Maximize γ under the constraint:

- $||w||^2 = 1$
- For all i, $y_i w \cdot x_i \geq \gamma$

objective function

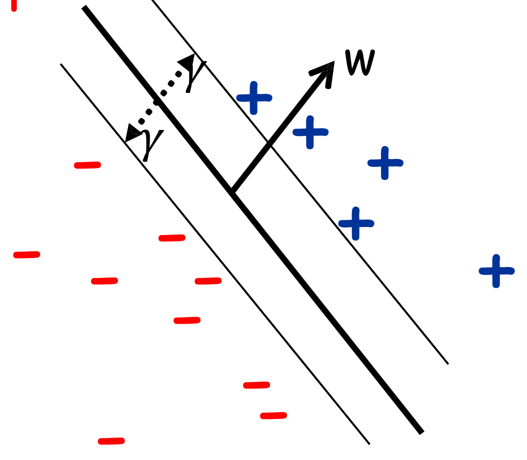
constraints

This is a **constrained optimization** problem.

- Famous example of constrained optimization: **linear programming**, where objective fn is linear, constraints are linear (in)equalities

Support Vector Machines (SVMs)

Directly optimize for the maximum margin separator: SVMs



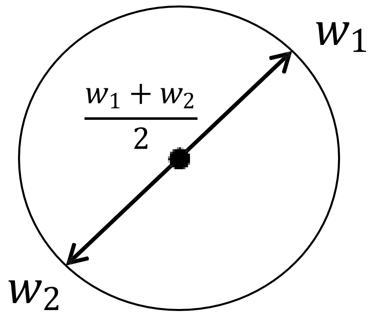
Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Maximize γ under the constraint:

- $\|w\|^2 = 1$
- For all i, $y_i w \cdot x_i \geq \gamma$

This constraint is non-linear.

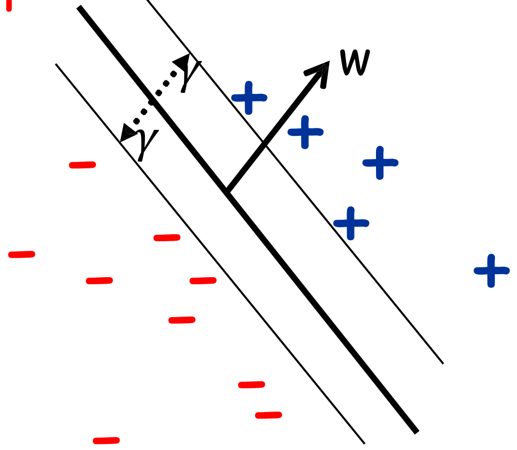
In fact, it's even non-convex



Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

Directly optimize for the maximum margin separator: SVMs

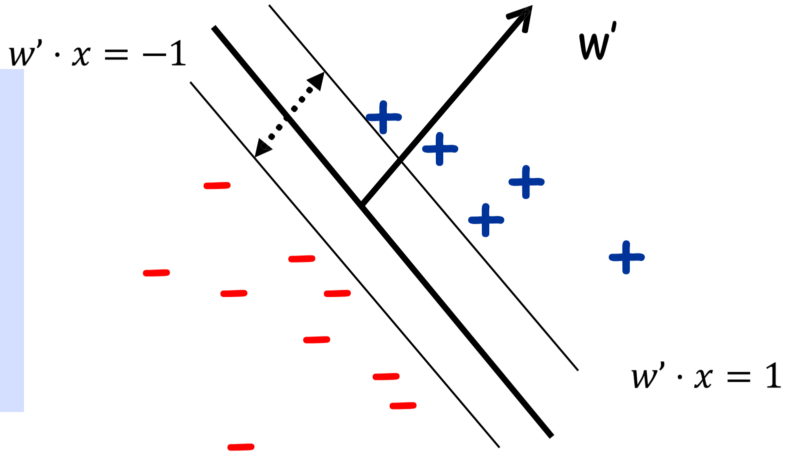


Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Maximize γ under the constraint:

- $||w||^2 = 1$
- For all i, $y_i w \cdot x_i \geq \gamma$

$w' = w/\gamma$, then max γ is equiv. to minimizing $||w'||^2$ (since $||w'||^2 = 1/\gamma^2$).
So, dividing both sides by γ and writing in terms of w′ we get:



Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Minimize $||w'||^2$ under the constraint:

- For all i, $y_i w' \cdot x_i \geq 1$

Support Vector Machines (SVMs)

Directly optimize for the maximum margin separator: SVMs

Input: S={(x_1, y_1), …,(x_m, y_m)};

$\mathrm{argmin}_w \; ||w||^2$ s.t.:

- For all i, $y_i w \cdot x_i \geq 1$

This is a **constrained optimization** problem.

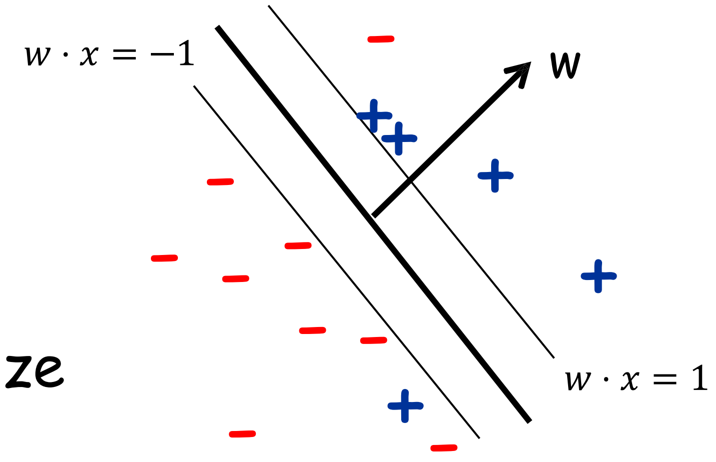
- The objective is convex (quadratic)
- All constraints are linear
- Can solve efficiently (in poly time) using standard **quadratic programing** (QP) software

Support Vector Machines (SVMs)

Question: what if data isn't perfectly linearly separable?

<u>Issue 1</u>: now have two objectives

- maximize margin
- minimize # of misclassifications.



Ans 1: Let's optimize their sum: minimize

$$||w||^2 + C(\text{\# misclassifications})$$

where C is some tradeoff constant.

<u>Issue 2</u>: This is computationally hard (NP-hard).

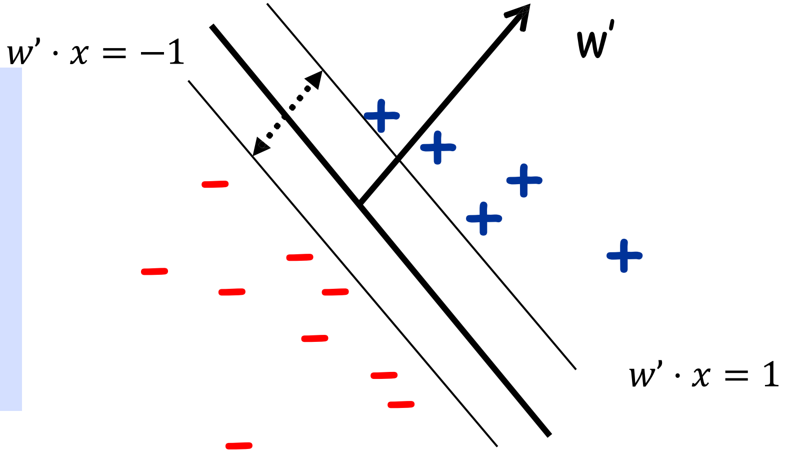
[even if didn't care about margin and minimized # mistakes]

NP-hard [Guruswami-Raghavendra'06]

Support Vector Machines (SVMs)

Question: what if data isn't perfectly linearly separable?

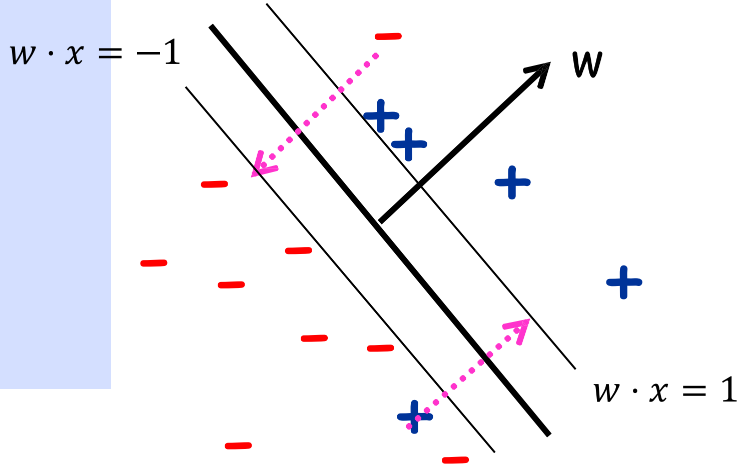
Replace "# mistakes" with upper bound called "hinge loss"



Input: S={(x_1, y_1), …,(x_m, y_m)};

Minimize $||w'||^2$ under the constraint:

- For all i, $y_i w' \cdot x_i \geq 1$



Input: S={(x_1, y_1), …,(x_m, y_m)};

Find $\mathrm{argmin}_{w,\xi_1,\ldots,\xi_m} ||w||^2 + C\sum_i \xi_i$ s.t.:

- For all i, $y_i w \cdot x_i \geq 1 - \xi_i$

  $\xi_i \geq 0$

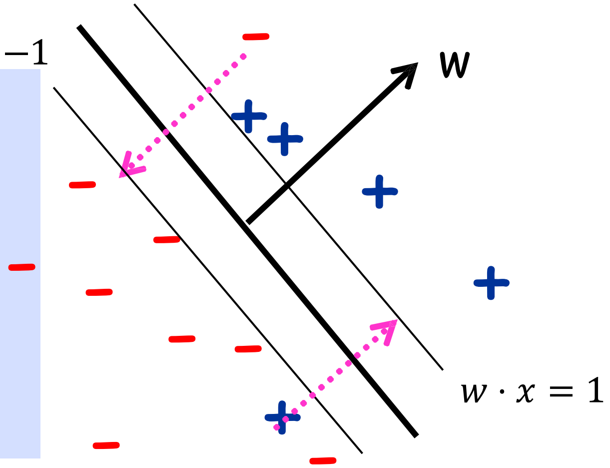
ξ_i are "slack variables"

Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

Question: what if data isn't perfectly linearly separable?

Replace "# mistakes" with upper bound called "hinge loss"



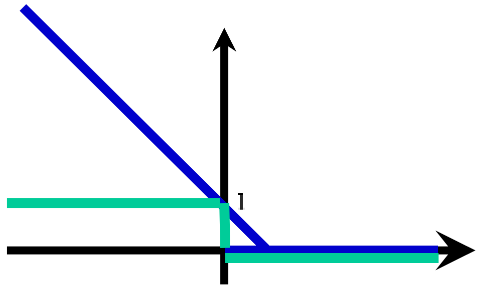
Input: S={(x_1, y_1), …,(x_m, y_m)};

Find $\mathrm{argmin}_{w,\xi_1,\ldots,\xi_m} ||w||^2 + C\sum_i \xi_i$ s.t.:

- For all i, $y_i w \cdot x_i \geq 1 - \xi_i$
  $\xi_i \geq 0$

ξ_i are "slack variables"

C controls the relative weighting between the twin goals of making the $||w||^2$ small (margin is large) and ensuring that most examples have functional margin ≥ 1.



$l(w,x,y) = \max(0, 1 - y\, w \cdot x)$

Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

Question: what if data isn't perfectly linearly separable?

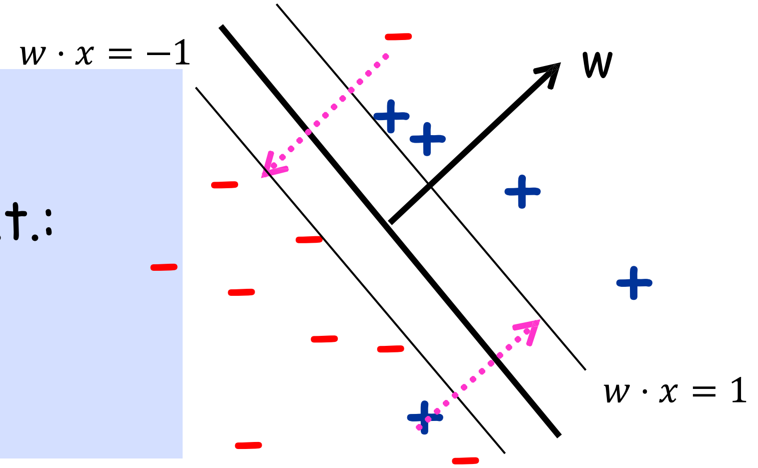
Replace "# mistakes" with upper bound called "hinge loss"

Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Find $\mathrm{argmin}_{w,\xi_1,\ldots,\xi_m} \; ||w||^2 + C\sum_i \xi_i$ s.t.:

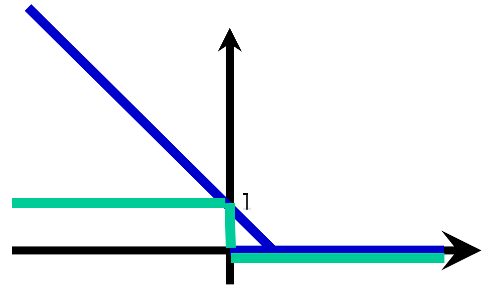
- For all i, $y_i w \cdot x_i \geq 1 - \xi_i$

  $\xi_i \geq 0$



Replace the number of mistakes with the hinge loss

$$||w||^2 + C(\#\ \text{misclassifications})$$



$l(w, x, y) = \max(0, 1 - y\, w \cdot x)$

Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

Question: what if data isn't perfectly linearly separable?

Replace "# mistakes" with upper bound called "hinge loss"

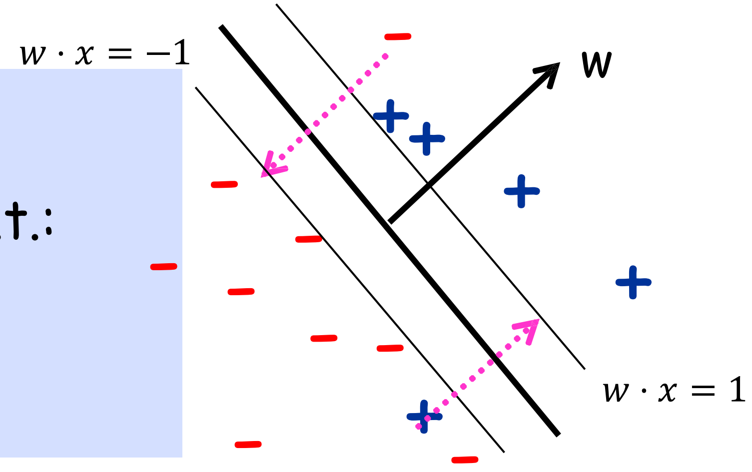
Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Find $\mathrm{argmin}_{w,\xi_1,\ldots,\xi_m} \; ||w||^2 + C\sum_i \xi_i$ s.t.:

- For all i, $y_i w \cdot x_i \geq 1 - \xi_i$

  $\xi_i \geq 0$

ξ_i are "slack variables"



Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

Question: what if data isn't perfectly linearly separable?

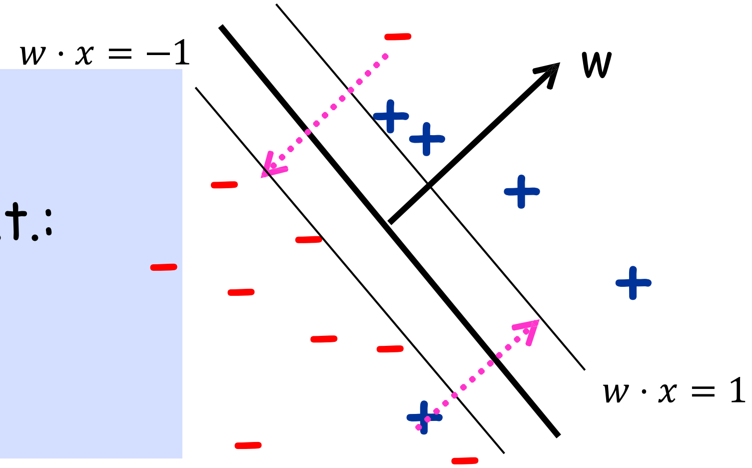
Replace "# mistakes" with upper bound called "hinge loss"

<u>Input</u>: S={(x_1, y_1), …,(x_m, y_m)};

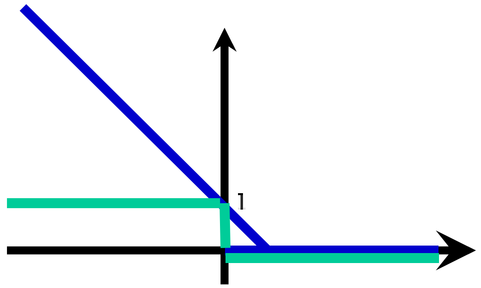
<u>Find</u> $\text{argmin}_{w,\xi_1,\ldots,\xi_m} \|w\|^2 + C\sum_i \xi_i$ s.t.:

- For all i, $y_i w \cdot x_i \geq 1 - \xi_i$

  $\xi_i \geq 0$



Total amount have to move the points to get them on the correct side of the lines $w \cdot x = +1/-1$, where the distance between the lines $w \cdot x = 0$ and $w \cdot x = 1$ counts as "1 unit".



$l(w, x, y) = \max(0, 1 - y\, w \cdot x)$

What if the data is far from being linearly separable?

Example:  vs 

No good linear separator in pixel representation.

SVM philosophy: "Use a Kernel"

Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

Input: S={(x_1, y_1), …,(x_m, y_m)};

Find $\text{argmin}_{w,\xi_1,\ldots,\xi_m} \|w\|^2 + C\sum_i \xi_i$ s.t.:

- For all i, $y_i w \cdot x_i \geq 1 - \xi_i$

  $\xi_i \geq 0$

Primal form

Which is equivalent to:

Input: S={(x_1, y_1), …,(x_m, y_m)};

Find $\text{argmin}_{\boldsymbol{\alpha}} \frac{1}{2}\sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i \cdot x_j - \sum_i \alpha_i$ s.t.:

- For all i, $0 \leq \alpha_i \leq C_i$

  $\sum_i y_i \alpha_i = 0$

Lagrangian Dual

SVMs (Lagrangian Dual)

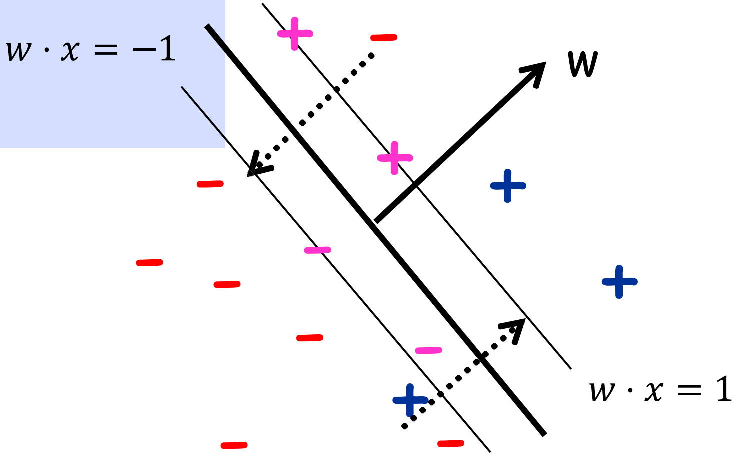
上海科技大学 ShanghaiTech University

Input: S={(x_1, y_1), …,(x_m, y_m)};

Find $\operatorname{argmin}_{\boldsymbol{\alpha}} \frac{1}{2}\sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i \cdot x_j - \sum_i \alpha_i$ s.t.:

- For all i, $0 \leq \alpha_i \leq C_i$

  $\sum_i y_i \alpha_i = 0$



- Final classifier is: $w = \sum_i \alpha_i y_i x_i$
- The points x_i for which $\alpha_i \neq 0$ are called the "support vectors"

Kernelizing the Dual SVMs

上海科技大学
ShanghaiTech University

Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$;

Find $\operatorname{argmin}_{\alpha} \frac{1}{2}\sum_i \sum_j y_i y_j \alpha_i \alpha_j x_i \cdot x_j - \sum_i \alpha_i$ s.t.:

- For all i, $0 \le \alpha_i \le C_i$

$$\sum_i y_i \alpha_i = 0$$

Replace $x_i \cdot x_j$ with $K(x_i, x_j)$.

- Final classifier is: $w = \sum_i \alpha_i y_i x_i$
- The points x_i for which $\alpha_i \neq 0$ are called the "support vectors"
- With a kernel, classify x using $\sum_i \alpha_i y_i K(x, x_i)$

Support Vector Machines (SVMs)

上海科技大学 ShanghaiTech University

- One of the most theoretically well motivated and practically most effective classification algorithms in machine learning.

- Directly motivated by Margins and Kernels!

What you should know

- The importance of margins in machine learning.
- The primal form of the SVM optimization problem
- The dual form of the SVM optimization problem.
- Kernelizing SVM.