Support Vector Machines (SVMs). Kernelizing SVMs

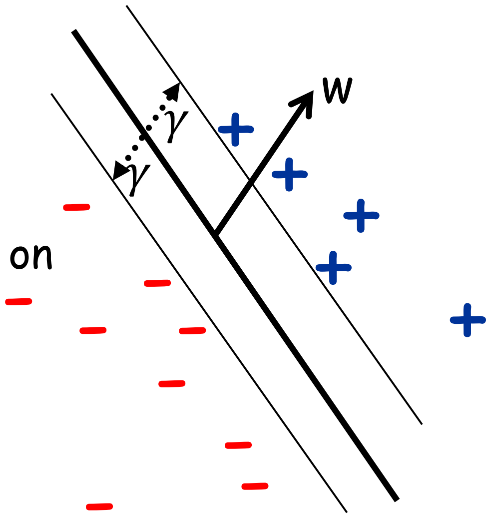
Maria-Florina Balcan

02/21/2018

Margin Important Theme in ML

- If large margin, # mistakes Peceptron makes is small (independent on the dim of the ambient space)!

- Large margin can help prevent overfitting.
  - 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].



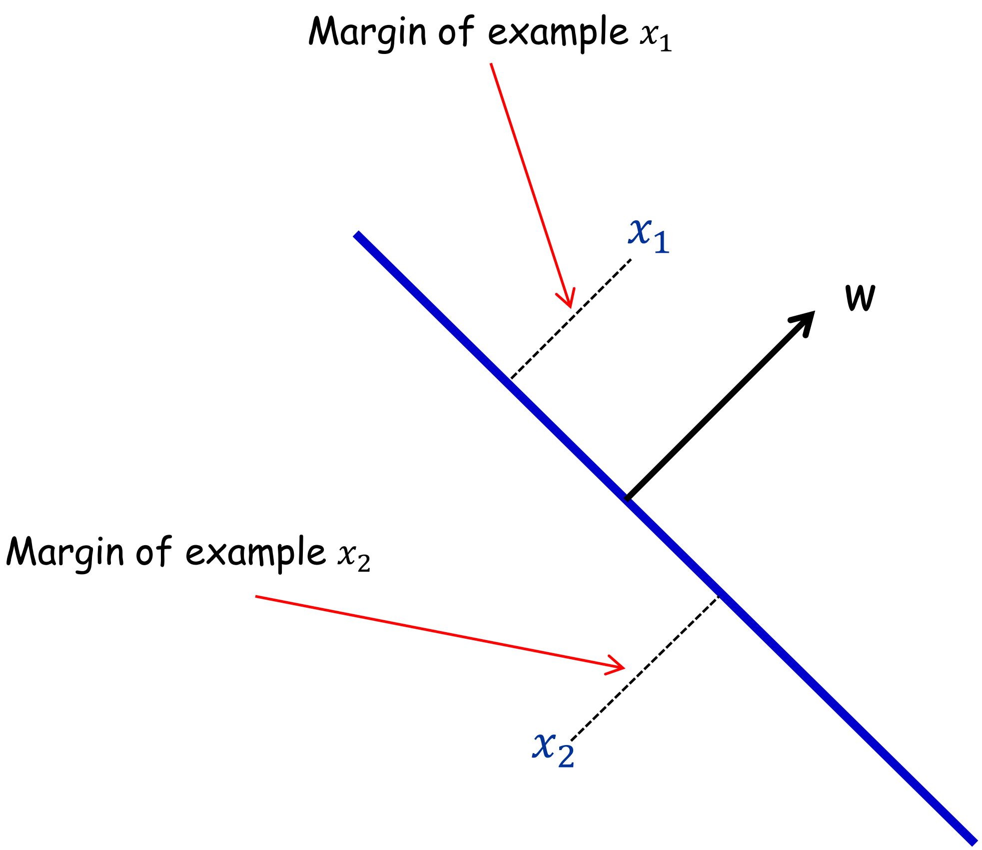
- Ideas: Directly search for a large margin classifier!!!

Support Vector Machines (SVMs).

Geometric Margin

WLOG homogeneous linear separators $[w_0 = 0]$.

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



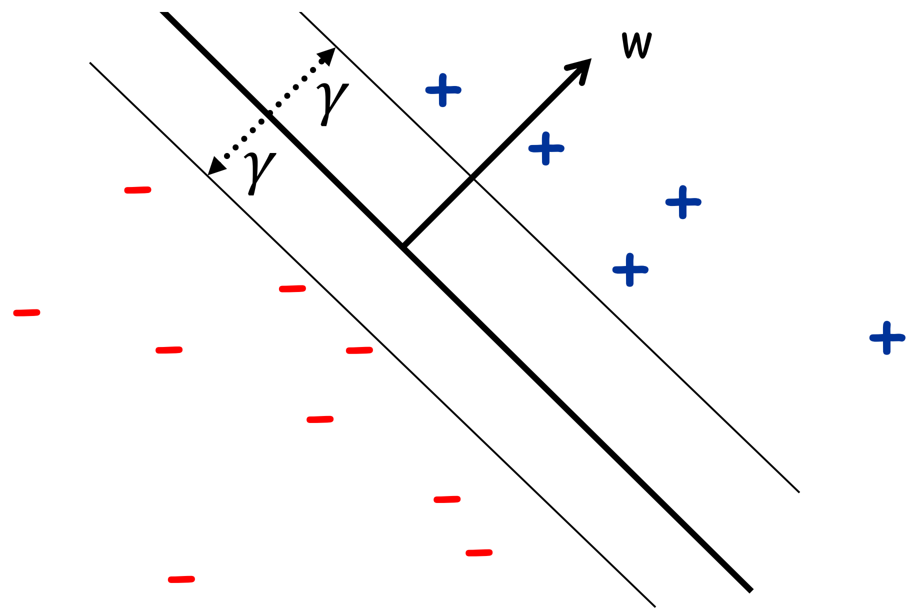
If $||w|| = 1$, margin of x w.r.t. w is $|x \cdot w|$.

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 $x \in S$.

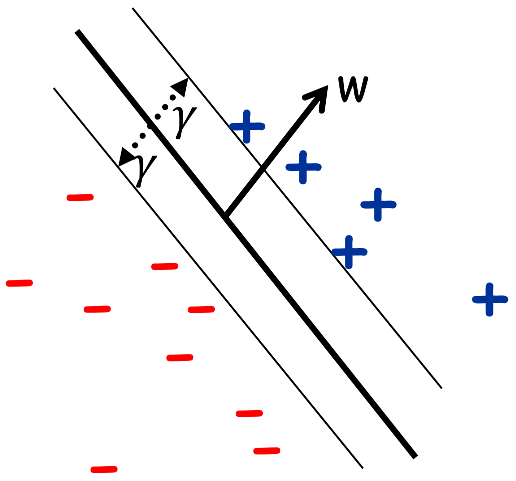
Definition: The margin γ of a set of examples S is the maximum γ_w over all linear separators w.



Support Vector Machines (SVMs)

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, $y_i w \cdot x_i \geq \gamma$

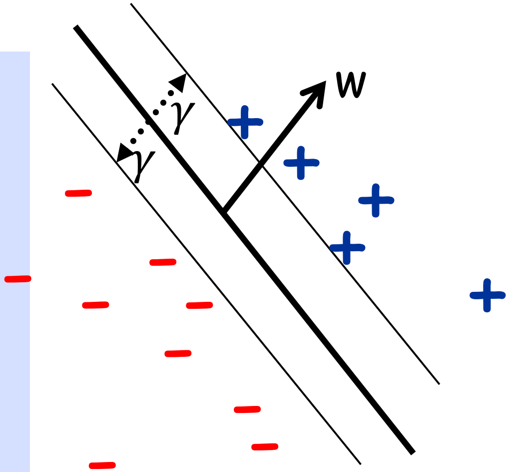
Output: w, a separator of margin γ over S

The case where the data is truly 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



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

Find: some w and maximum γ where:

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

Output: maximum margin separator over S

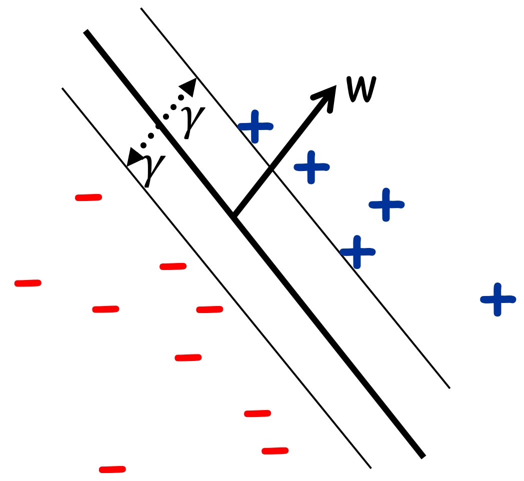
Support Vector Machines (SVMs)

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)

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

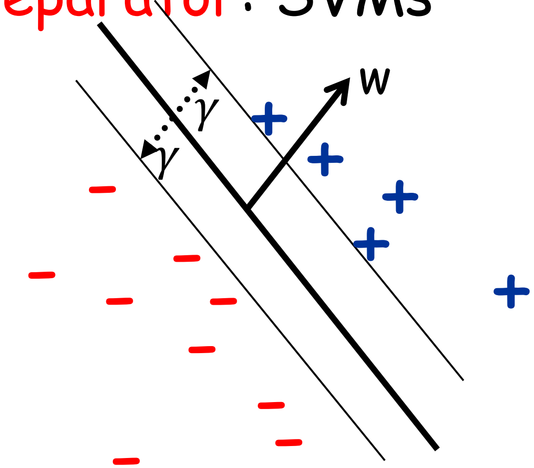
constraints

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



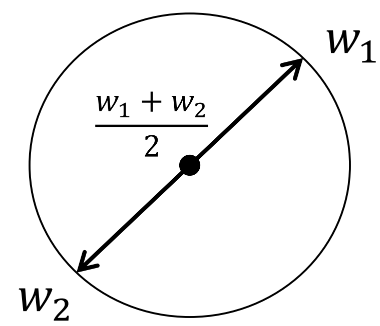
Input: S={(x_1, y_1), …,(x_m, y_m)};

Maximize γ under the constraint:

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

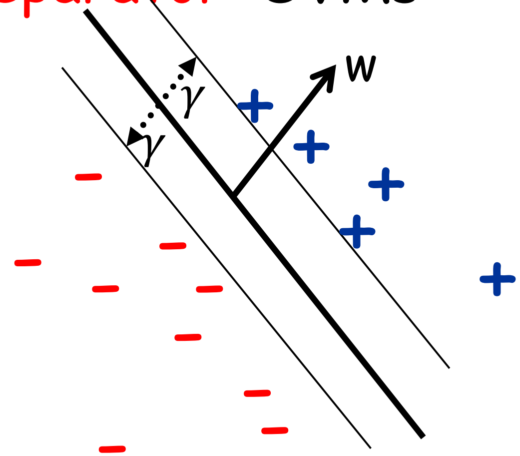
This constraint is non-linear.

In fact, it's even non-convex



Support Vector Machines (SVMs)

Directly optimize for the maximum margin separator: SVMs

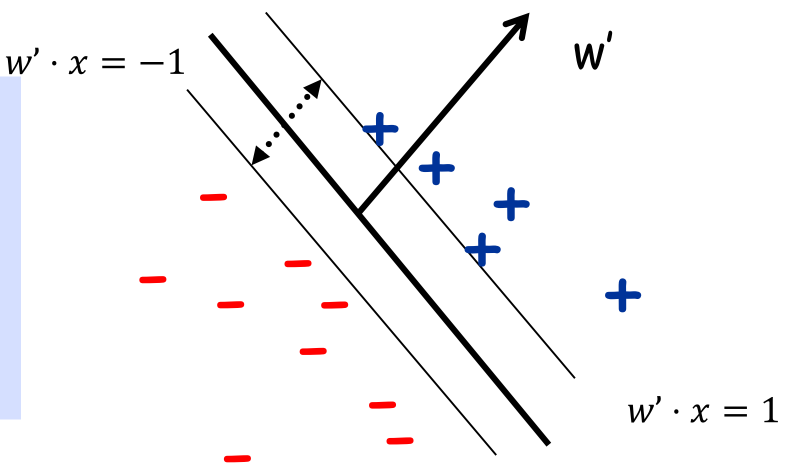


Input: S={(x_1, y_1), …,(x_m, y_m)};

Maximize γ under the constraint:

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

$w' = w/\gamma$, 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), …,(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)};

$\text{argmin}_w \, ||w||^2$ s.t.:

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

This is a **constrained optimization** problem.

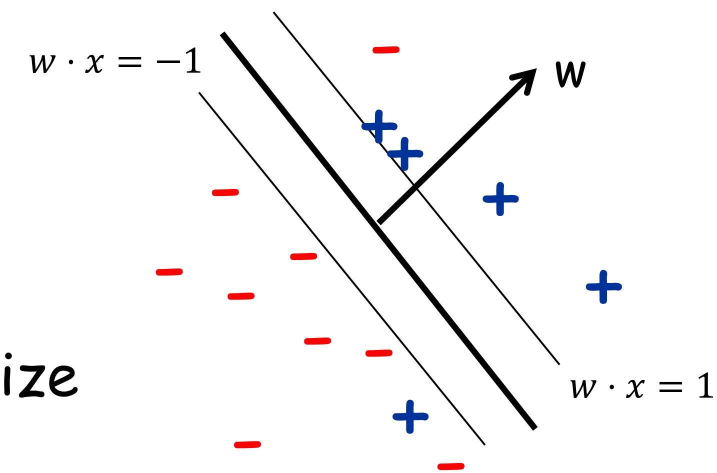
- The objective is convex (quadratic)
- All constraints are linear
- Can solve efficiently (in poly time) using standard **quadratic programing** (QP) software

Support Vector Machines (SVMs)

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

Issue 1: 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.

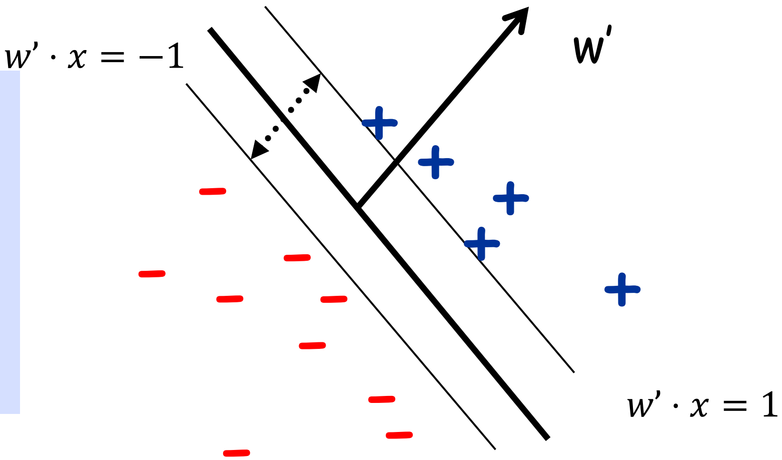
Issue 2: This is computationally very hard (NP-hard).

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

Support Vector Machines (SVMs)

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

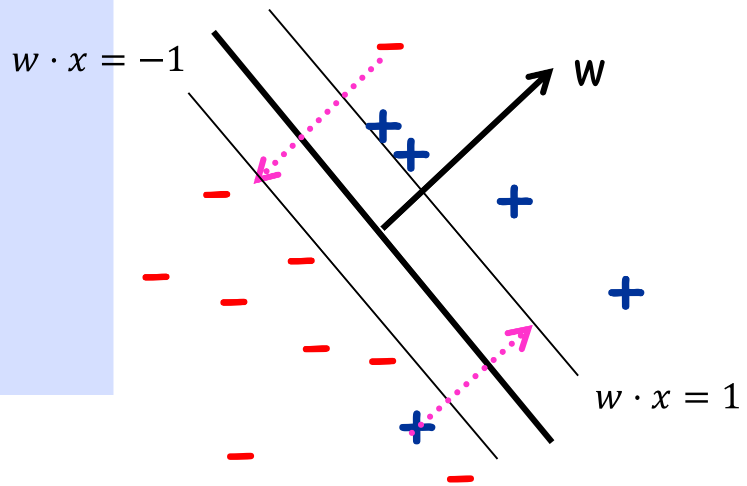
Replace "# mistakes" with upper bound called "hinge loss"



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

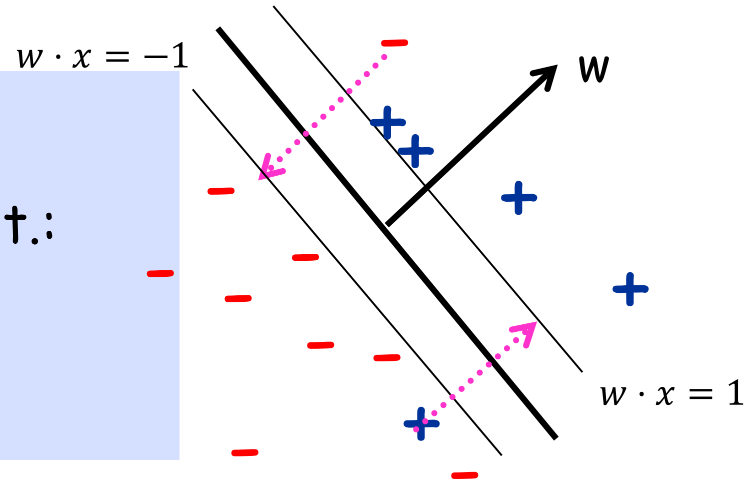
- For all i, $y_i w \cdot x_i \geq 1 - \xi_i$

  $\xi_i \geq 0$

ξ_i are "slack variables"

Support Vector Machines (SVMs)

Question: what if data isn't perfectly linearly separable?
Replace "# mistakes" with upper bound called "hinge loss"



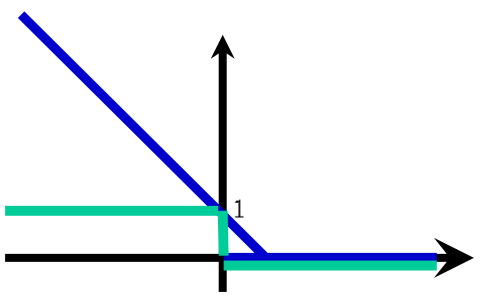
<u>Input</u>: S={(x_1, y_1), …,(x_m, y_m)};

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

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

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.

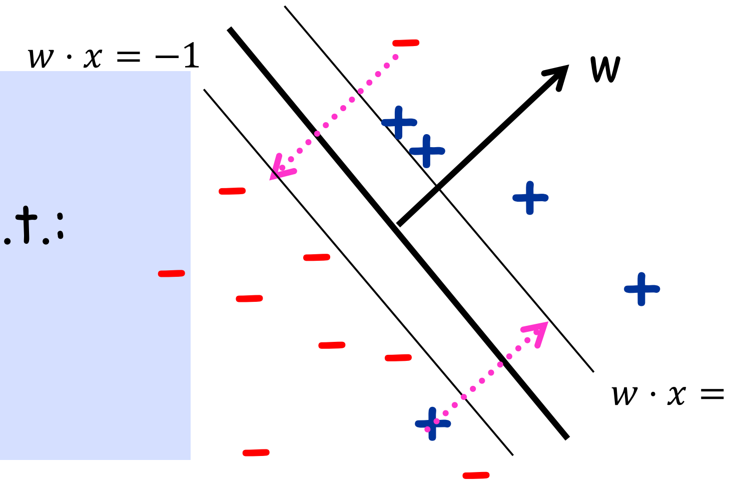


$l(w, x, y) = \max(0, 1 - y\, w \cdot x)$

Support Vector Machines (SVMs)

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

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



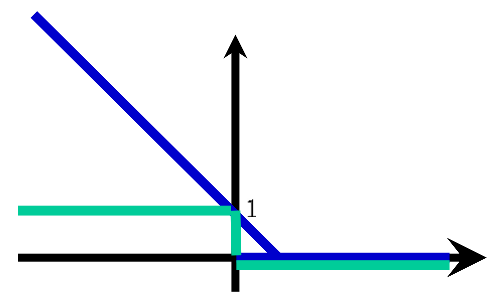
Input: S={(x_1, y_1), …,(x_m, y_m)};

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



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

Support Vector Machines (SVMs)

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, $y_i w \cdot x_i \geq 1 - \xi_i$

$$\xi_i \geq 0$$

Primal form

Which is equivalent to:

Can be kernelized!!!

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

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

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

$$\sum_i y_i \alpha_i = 0$$

Lagrangian Dual

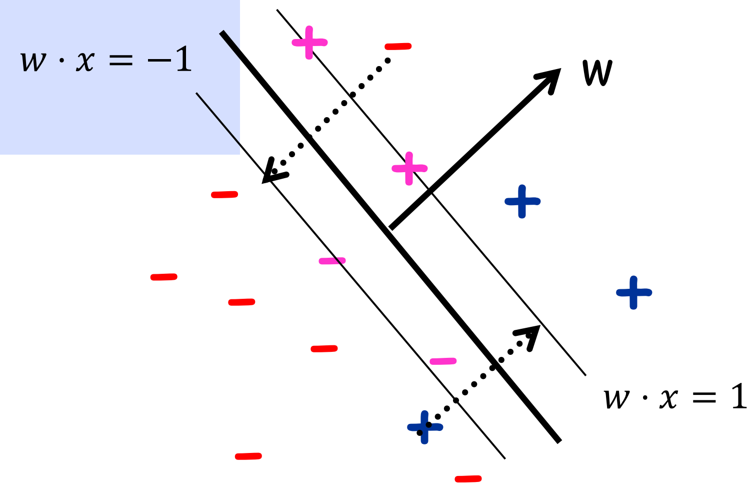
SVMs (Lagrangian Dual)

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

<u>Find</u> $\mathrm{argmin}_\alpha \frac{1}{2} \sum_i \sum_j y_i y_j \, \alpha_i \alpha_j x_i \cdot x_j - \sum_i \alpha_i$ s.t.:

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

$$\sum_i y_i \alpha_i = 0$$



- 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"

What you should know

- The importance of margins in machine learning.
- The SVM algorithm. Primal and Dual Form.
- Kernelizing SVM.