Machine Learning and Data Mining

Lecture 4.2: Hard-Margin Support Vector Machines

CS 434

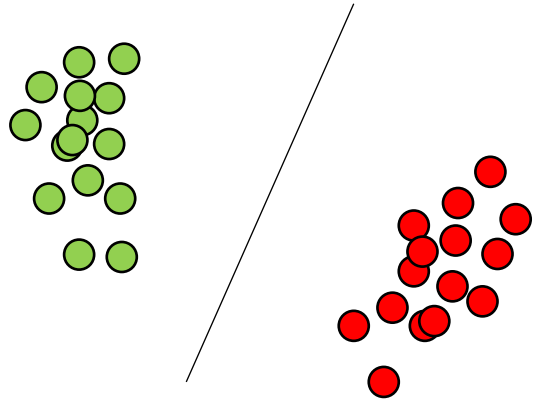
RECAP

From Last Lecture

Generative vs. Discriminative Models

Discriminative Classifiers:

- Learn $P(y|\boldsymbol{x})$ directly
- Logistic regression is one example
- *Nomenclature note -- people will also refer to algorithms that model no distribution as discriminative (such as kNN).*

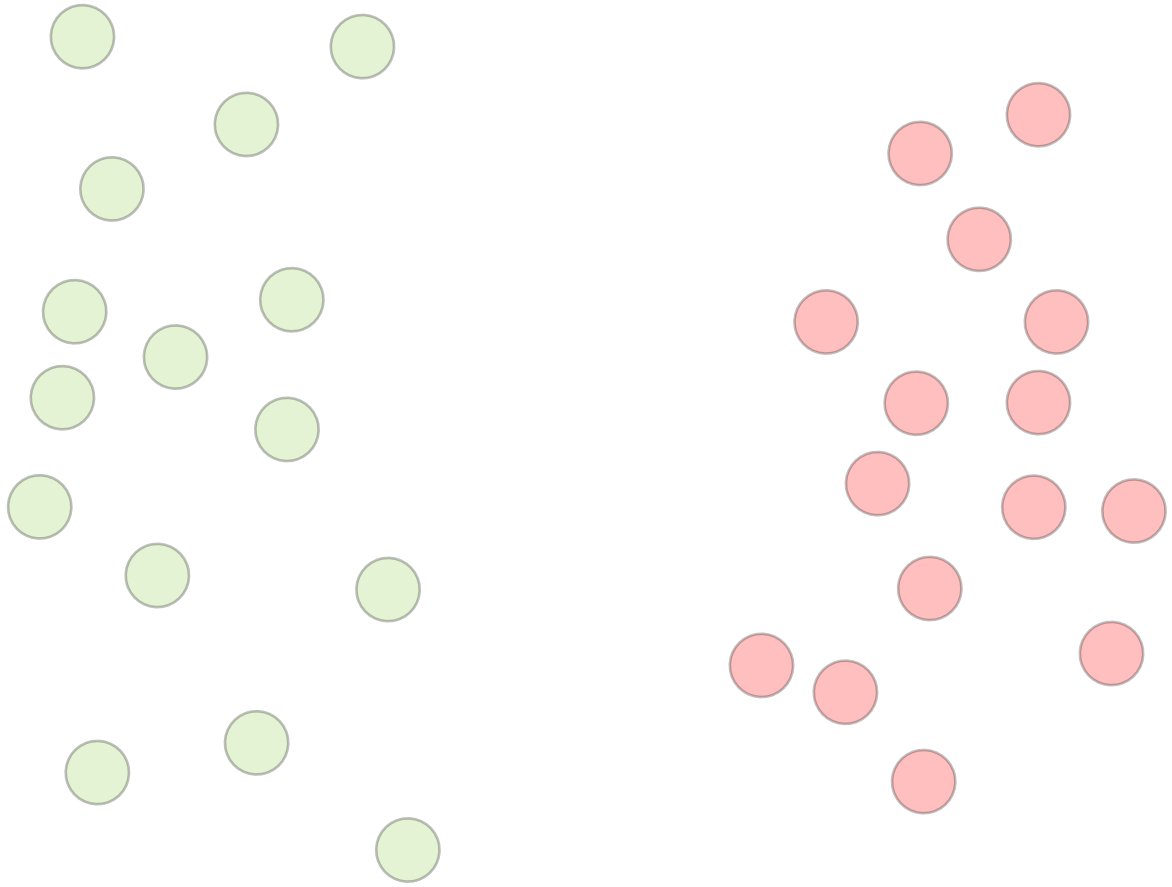


Today's Learning Objectives

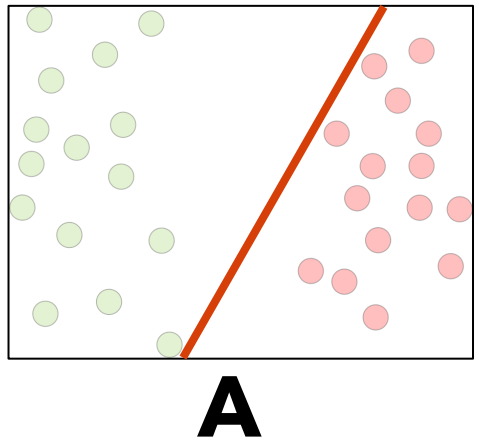
Be able to answer:

- What is the margin of a classifier?
- What is a Support Vector Machine?
  - How is a hard-margin SVM formalized?
  - What is the dual formalization?
  - What is its decision boundary?
  - What is a support vector?
- We'll have to touch on a few math tools to do this:
  - Constrained Optimization with Lagrange Multipliers
  - Quadratic Programming Solvers
  - You do not need to have an operational understanding of these two techniques at all for this course.

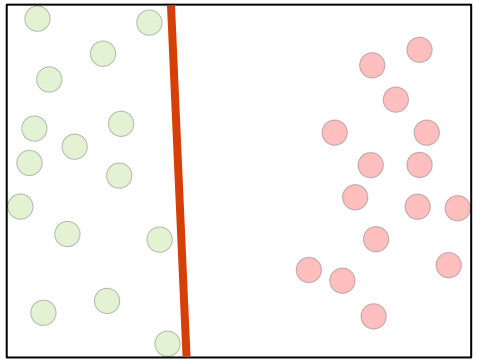
A Motivating Discussion



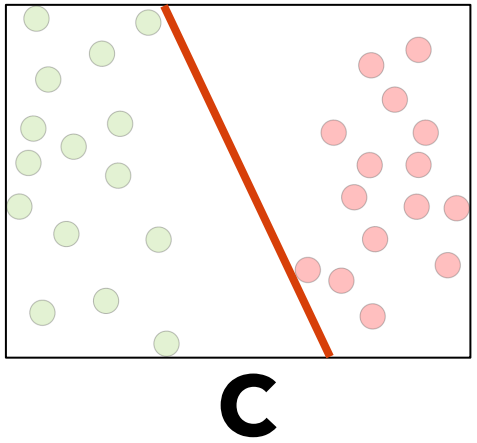
Question Break!



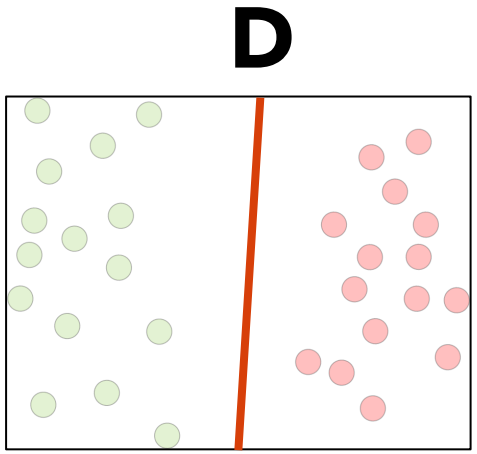
A



B



C



D

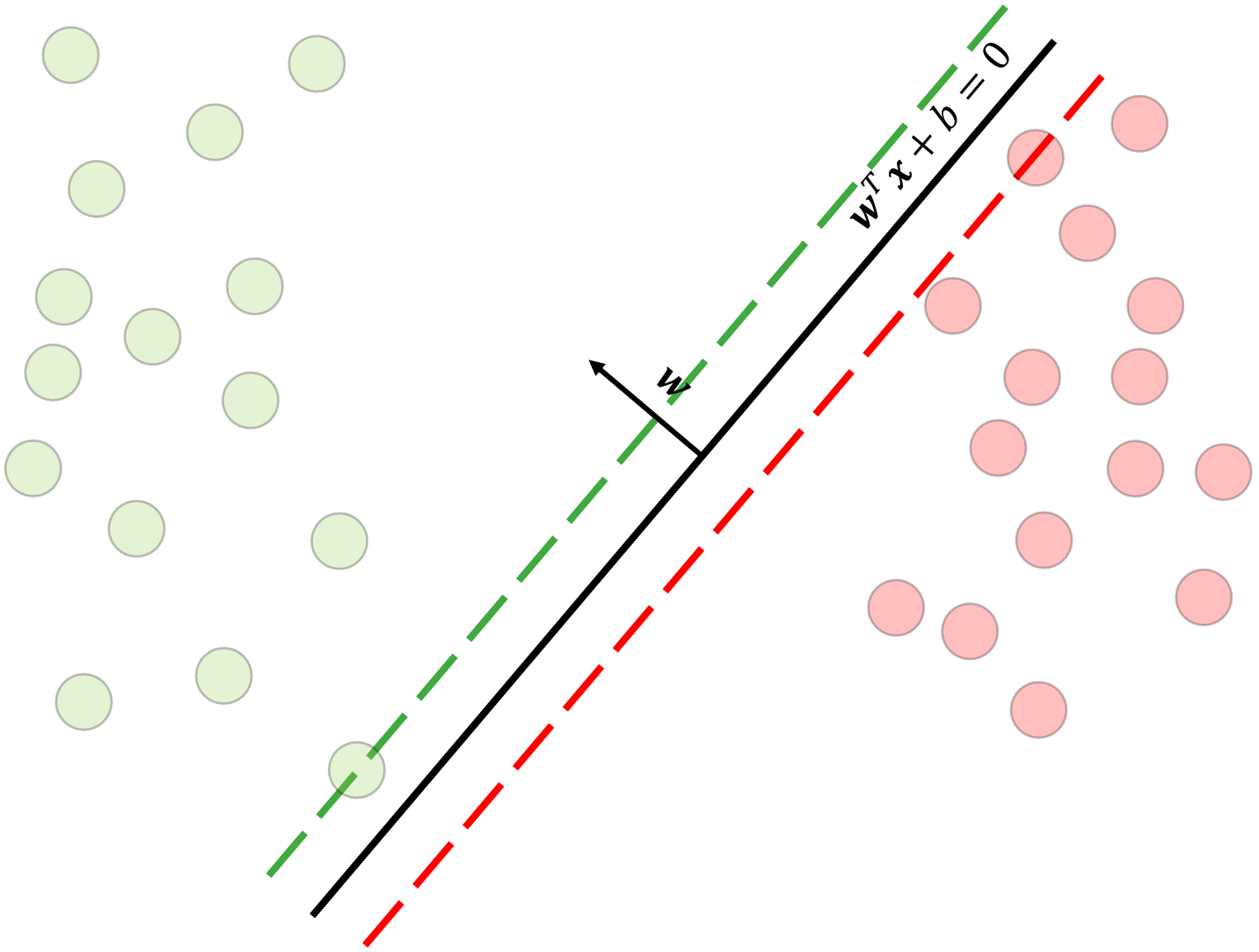
A I prefer A

B I prefer B

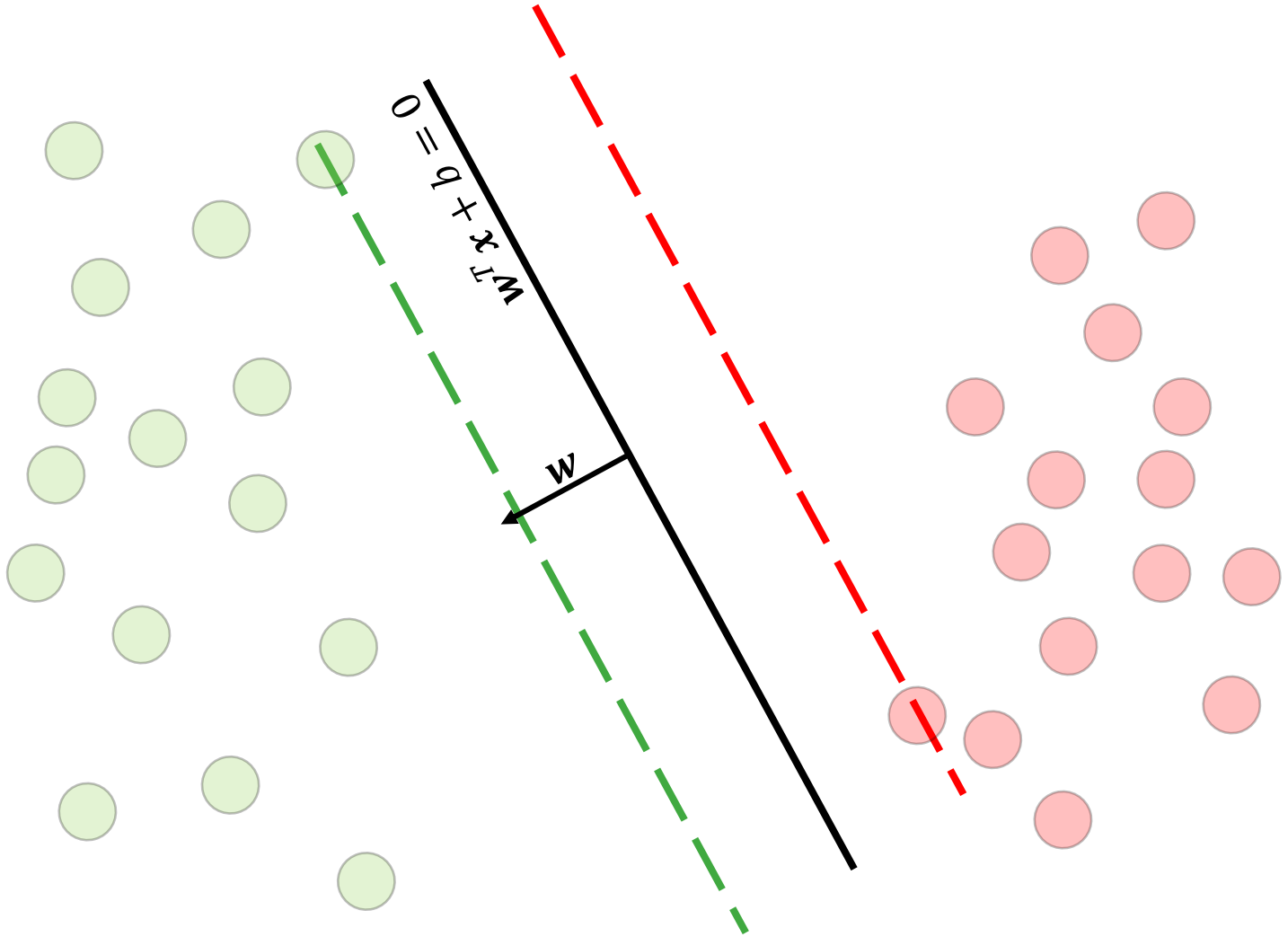
C I prefer C

D I prefer D

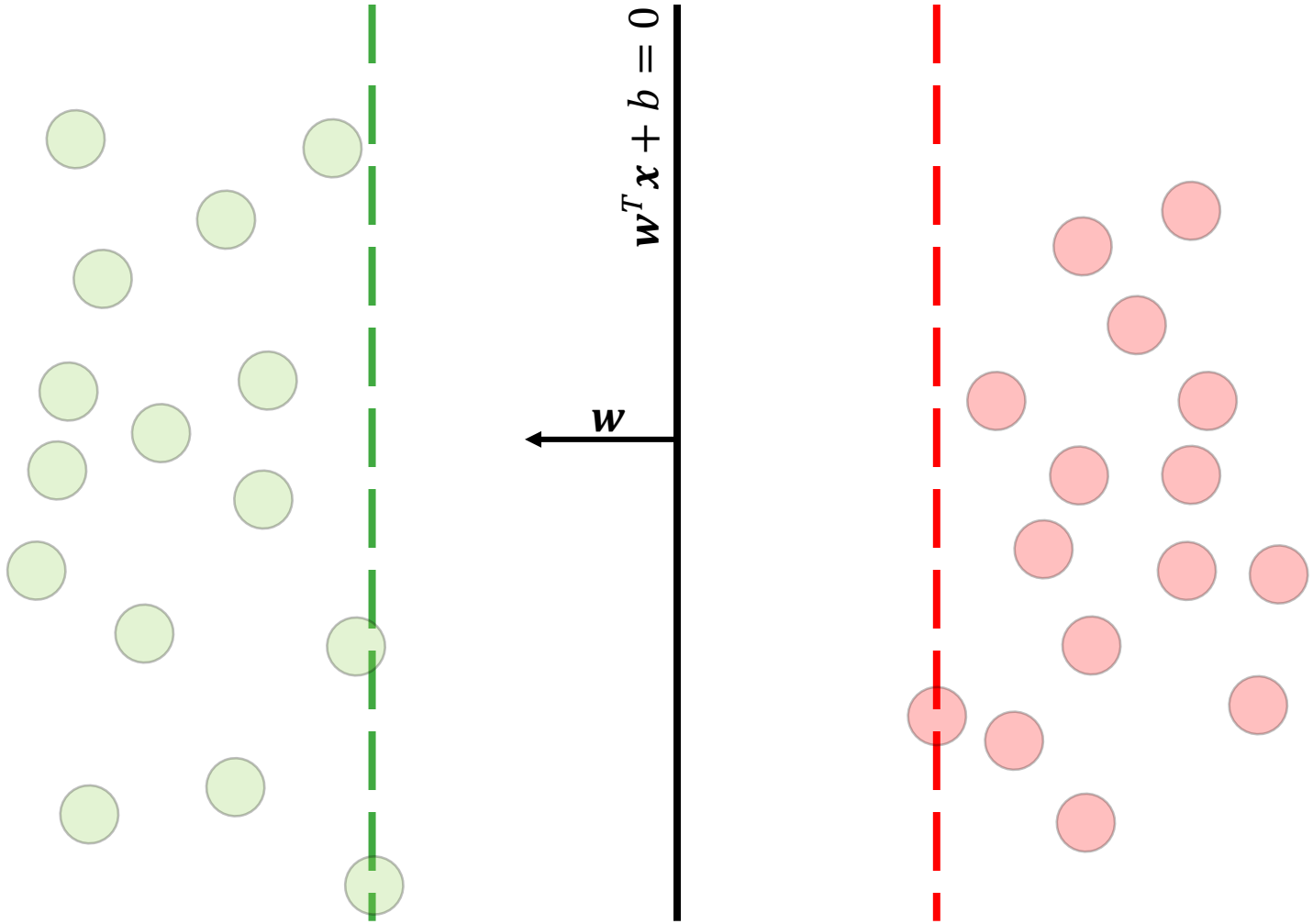
A Motivating Discussion



A Motivating Discussion



A Motivating Discussion



New Topic: Support Vector Machines

Algorithm Name: Support Vector Machines (SVM)
Type: Supervised Linear Classifier

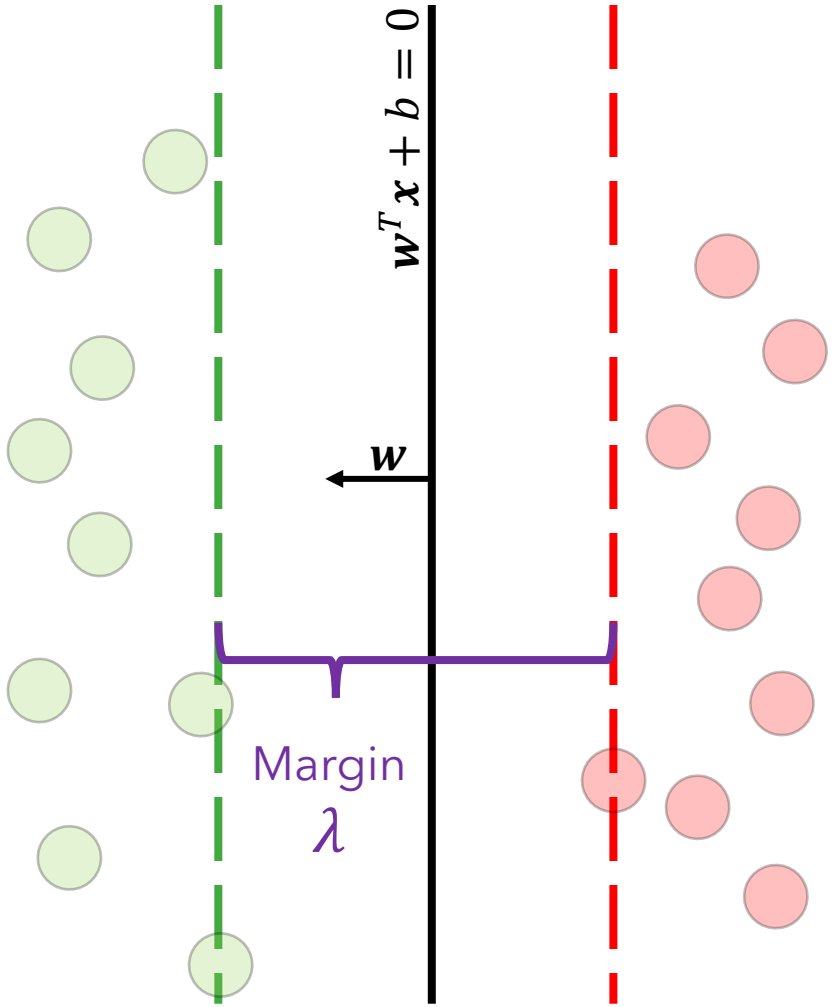
Key Idea: Find the linear decision boundary that maximizes the margin between positive and negative examples.

What is a support vector? We'll need to run through a lot of somewhat unintuitive math to really understand the structure of this problem. But the result will give us tremendous insight into this problem … and it'll build character.

Web Demo

https://jgreitemann.github.io/svm-demo

Max-Margin Classifiers



Our goal for today:

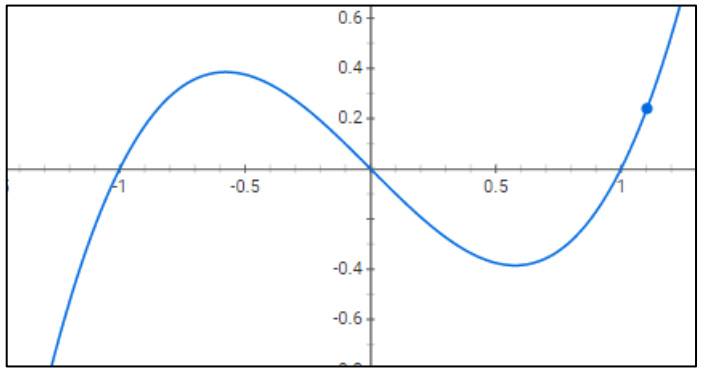
We want to find a classifier that separates the classes while maximizing the margin between positive and negative examples. *For now, we'll assume linear separability and fix that later.*

$$\max_{w,b} \quad \text{margin}$$

$$s.t. \quad \text{all examples are correct}$$

Question Break!

Provide a solution to the following constrained optimization problem:



$$\min_x f(x) = x^3 - x$$
$$s.t. \;\; 0 \le x \le 5$$

$$3x^2 - 1 = 0$$
$$\Rightarrow x = \pm\sqrt{0.33} \Rightarrow x = \sqrt{0.33}$$

$$x = 0 \Rightarrow f(0) = 0$$
$$x = 5 \Rightarrow f(5) = 120$$
$$x = \sqrt{0.33} \Rightarrow f(0.57) \approx -0.38$$

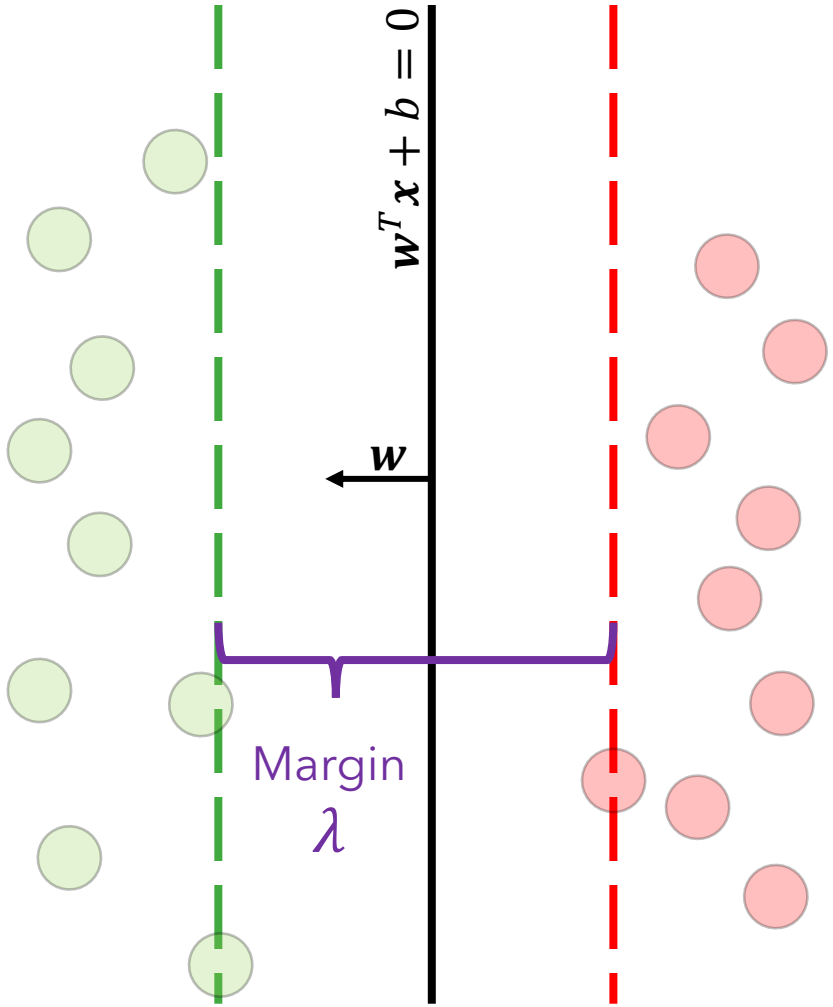
A 0

B 5

C $\sqrt{0.33}$

D $-\sqrt{0.33}$

Max-Margin Classifiers



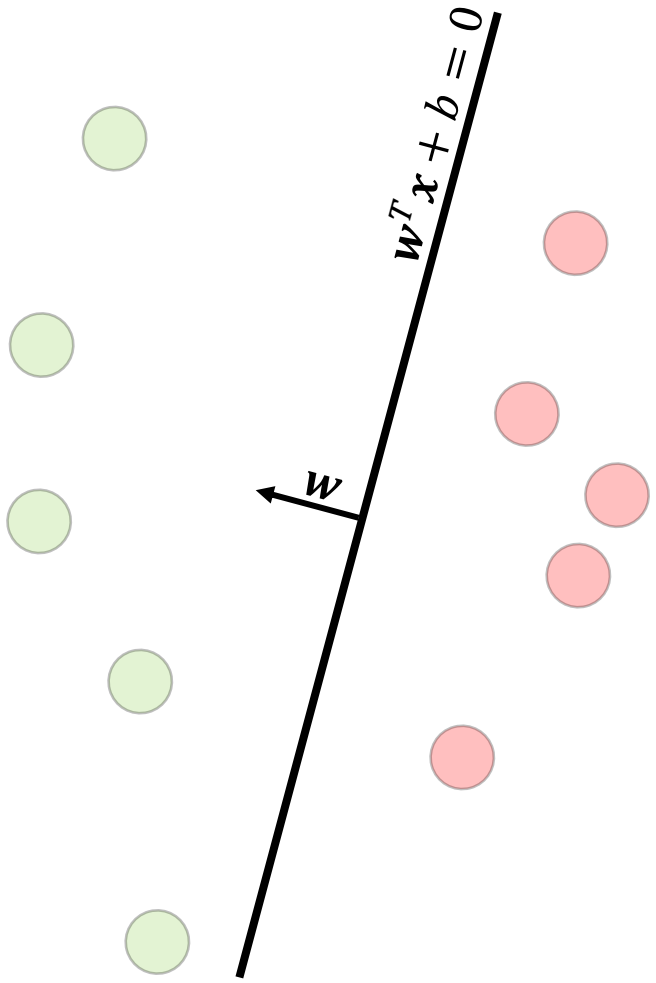
Our goal for today:

We want to find a classifier that separates the classes while maximizing the margin between positive and negative examples. *For now, we'll assume linear separability and fix that later.*

Our plan of attack:

1. Set up some convenient notation
2. Derive an expression for our intuition
3. Set up a constrained optimization problem
4. Solve it to arrive at a Support Vector Machine

(some new math tools to analyze this– Lagrange multipliers)



Problem Setup and Decision Rule

Consider a binary classification setting where we represent the classes as $y \in \{-1, 1\}$

Let our linear decision rule be:

$$\boldsymbol{w}^T \boldsymbol{x} + b > 0 \quad \Rightarrow \quad predict\ 1$$

$$\boldsymbol{w}^T \boldsymbol{x} + b < 0 \quad \Rightarrow \quad predict\ -1$$

Question Break!

I can tell if a point $\boldsymbol{x_i}$ with label $\boldsymbol{y_i}$ is correctly or incorrectly classified by checking if:

$$y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) > 0$$

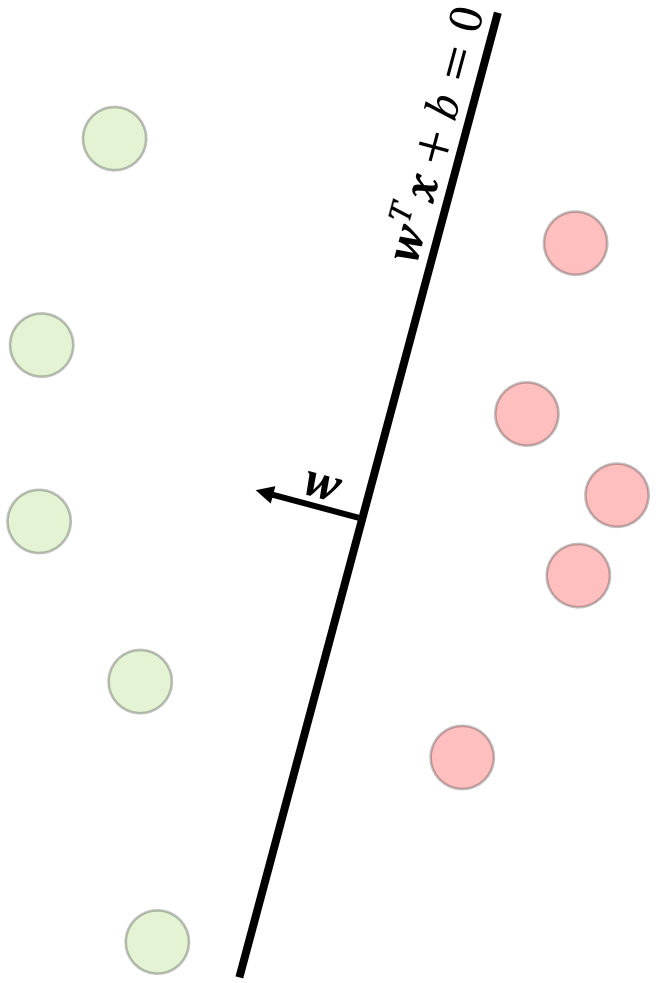
A True

B False

C I don't know

D

Step 1) Establish Some Convenient Notation



Problem Setup and Decision Rule

Consider a binary classification setting where we represent the classes as $y \in \{-1,1\}$

Let our linear decision rule be:

$$\boldsymbol{w}^T \boldsymbol{x} + b > 0 \quad \Rightarrow \quad predict\ 1$$

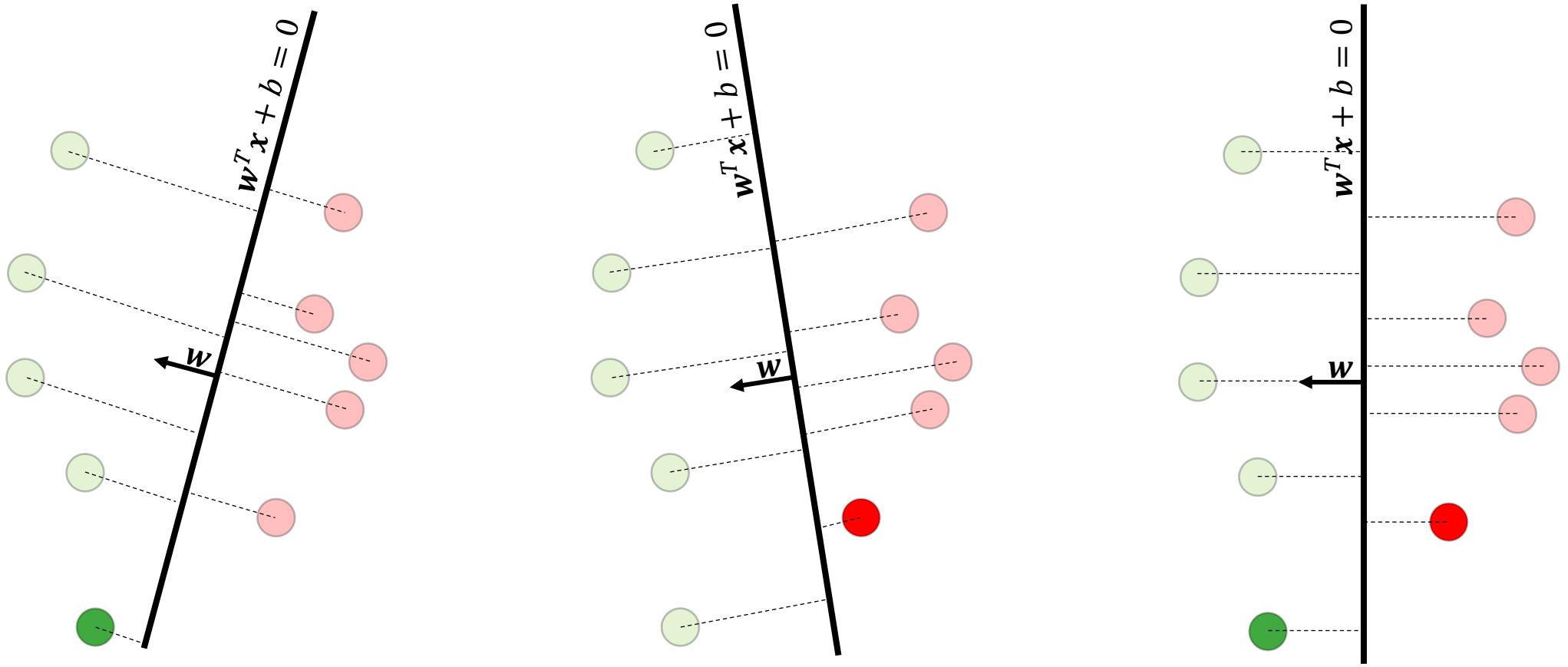
$$\boldsymbol{w}^T \boldsymbol{x} + b < 0 \quad \Rightarrow \quad predict\ -1$$

Means a correct classification has:

$$y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b) > 0$$

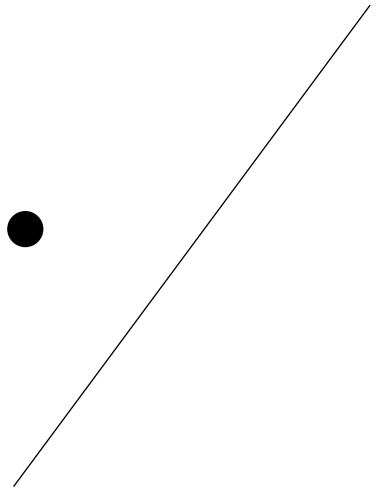
Step 2) Derive an Expression for the Margin

Intuition: One way to talk about the margin of our classifier would be to measure how close the nearest point is to the boundary.



Question Break!

The shortest path from a point to a line is perpendicular to the original line and parallel to the vector of weights defining it.



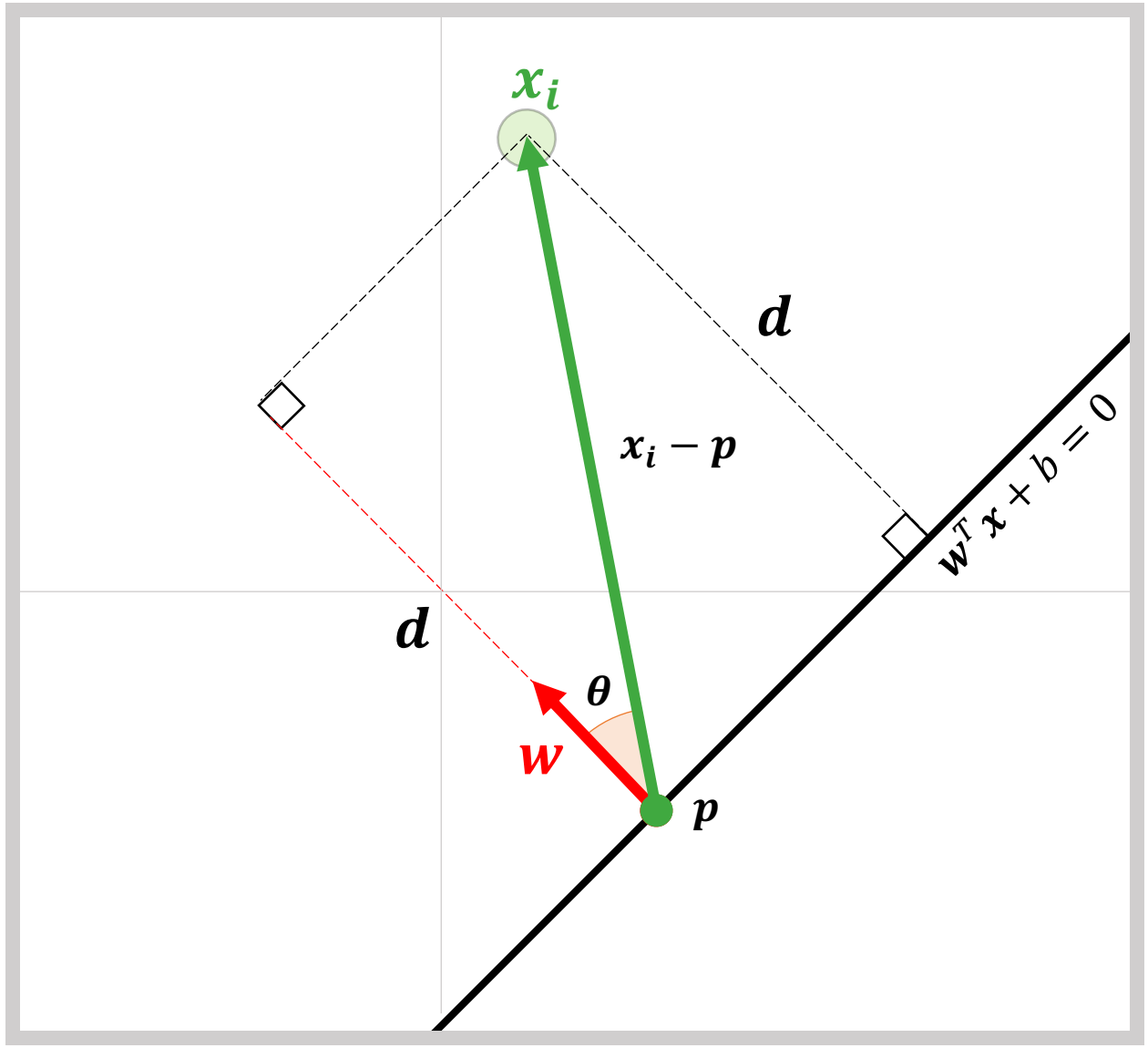
A True

B False

C I don't know

D I don't understand the question.

Step 2) Derive an Expression for the Margin



Goal: Find distance d.

Cos → adjacent over hypotenuse:

$$\cos(\theta) = \frac{d}{||x_i - p||}$$

Definition of dot product:

$$\frac{w^T(x_i - p)}{\|w\|\|x_i - p\|} = \frac{d}{||x_i - p||}$$

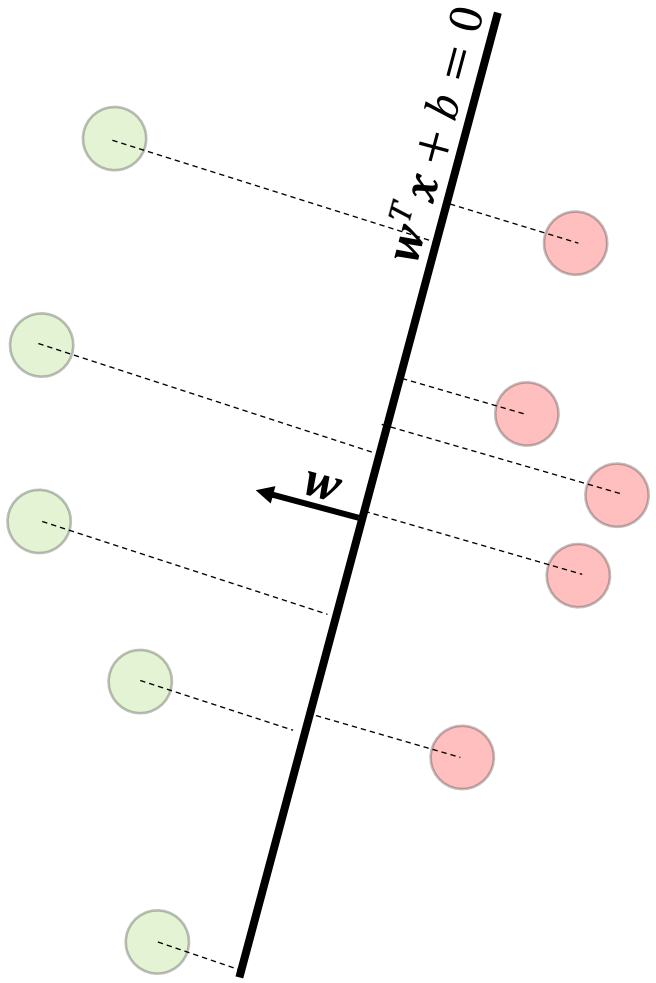
Point p is on the line so $w^T p = -b$

$$d = \frac{w^T(x_i - p)}{\|w\|} = \frac{w^T x_i + b}{\|w\|}$$

Absolute value to remove sign:

$$d = \frac{|w^T x_i + b|}{\|w\|}$$

Step 3) Set Up A Constrained Optimization Problem



We can then write the distance to the decision boundary from the nearest point as:

$$d_{min} = \min_i \frac{|w^T x_i + b|}{\|w\|}$$

One thing we could do is say we want to maximize this minimum distance without making any errors:

Maximize distance to nearest point

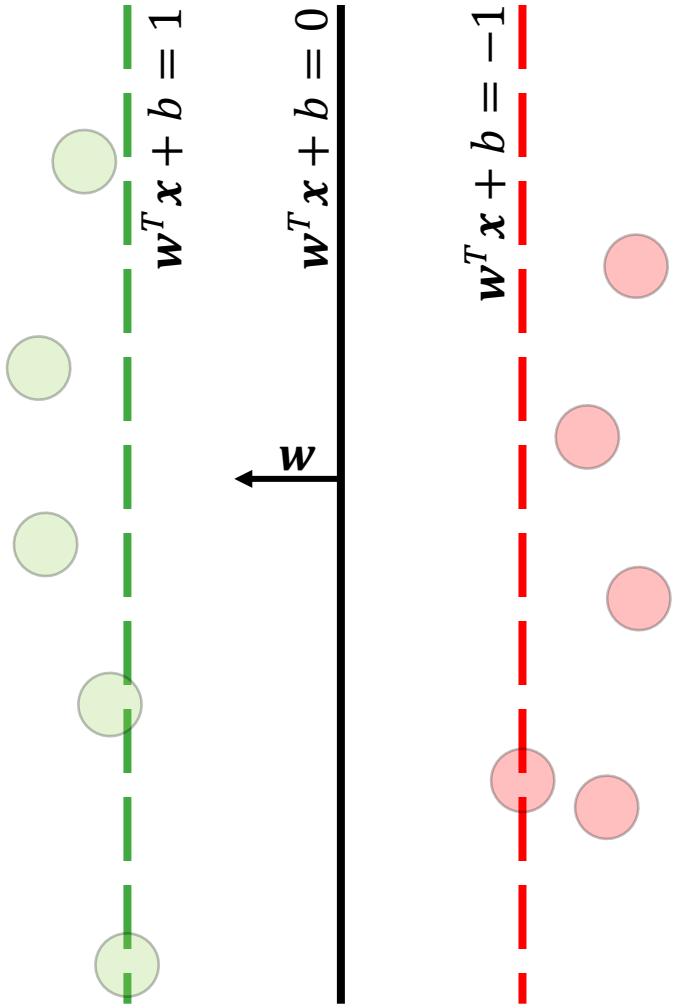
$$\max_{w,b} \min_i \frac{|w^T x_i + b|}{\|w\|}$$

While not making any errors

$$s.t.\ y_i(w^T x_i + b) > 0 \quad \forall i$$

This is an ugly problem! Max of min? w's everywhere?

Step 3) Set Up A Constrained Optimization Problem



To avoid dealing with this, let's constrain the minimum distance to some constant w.r.t to x – say $1/\|w\|$:

$$d_{min} = \min_i \frac{|w^T x_i + b|}{\|w\|} = \frac{1}{\|w\|} \quad \Rightarrow \quad |w^T x_i + b| \geq 1 \ \ \forall i$$

Can combine with "correctness" constraint $y_i(w^T x_i + b) > 0$:

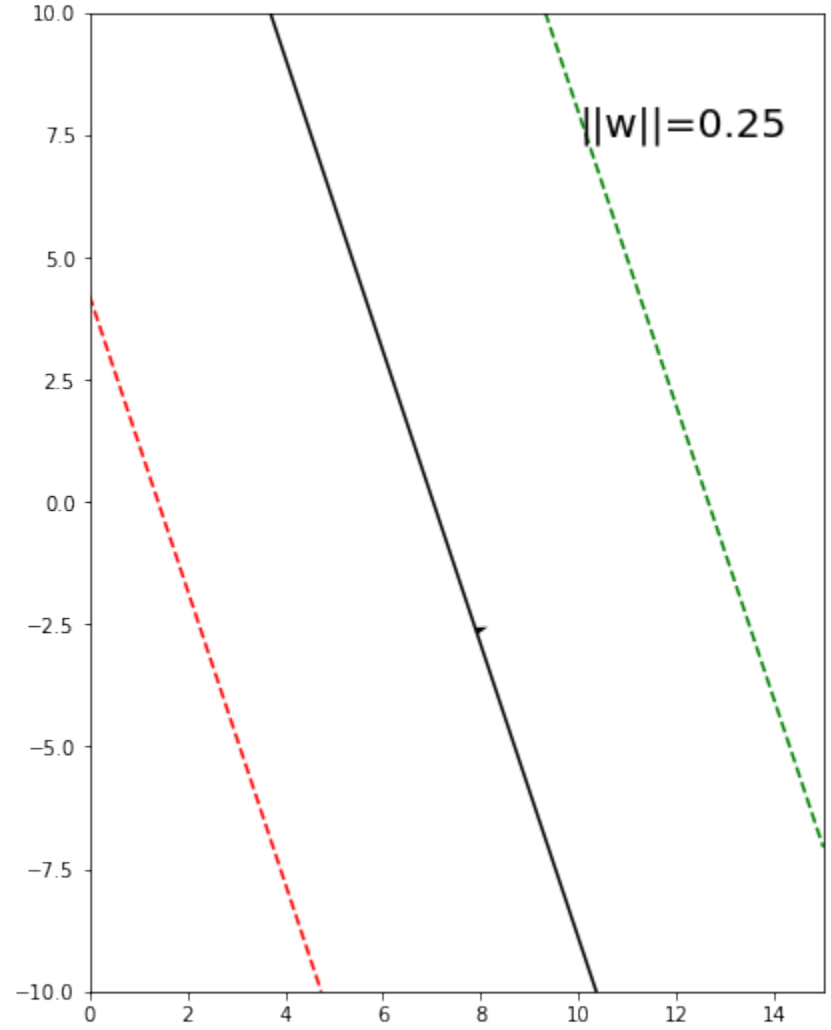
$$\Rightarrow \ y_i(w^T x_i + b) \geq 1 \ \forall i$$

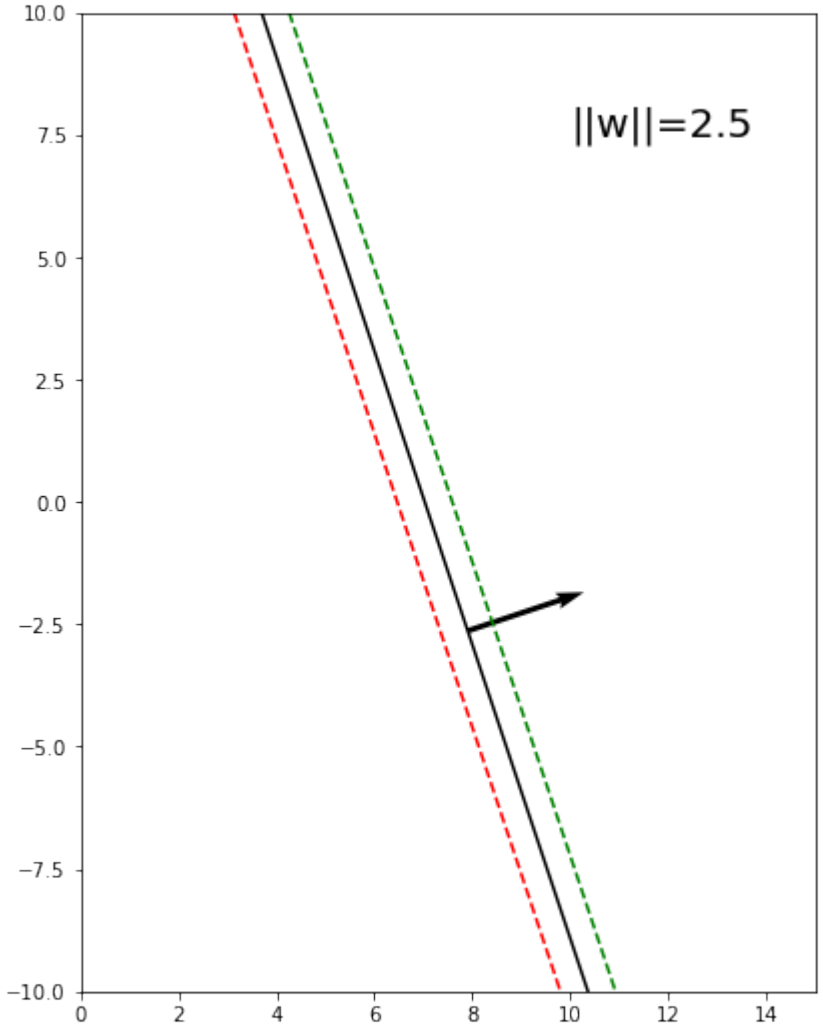
What does this do?

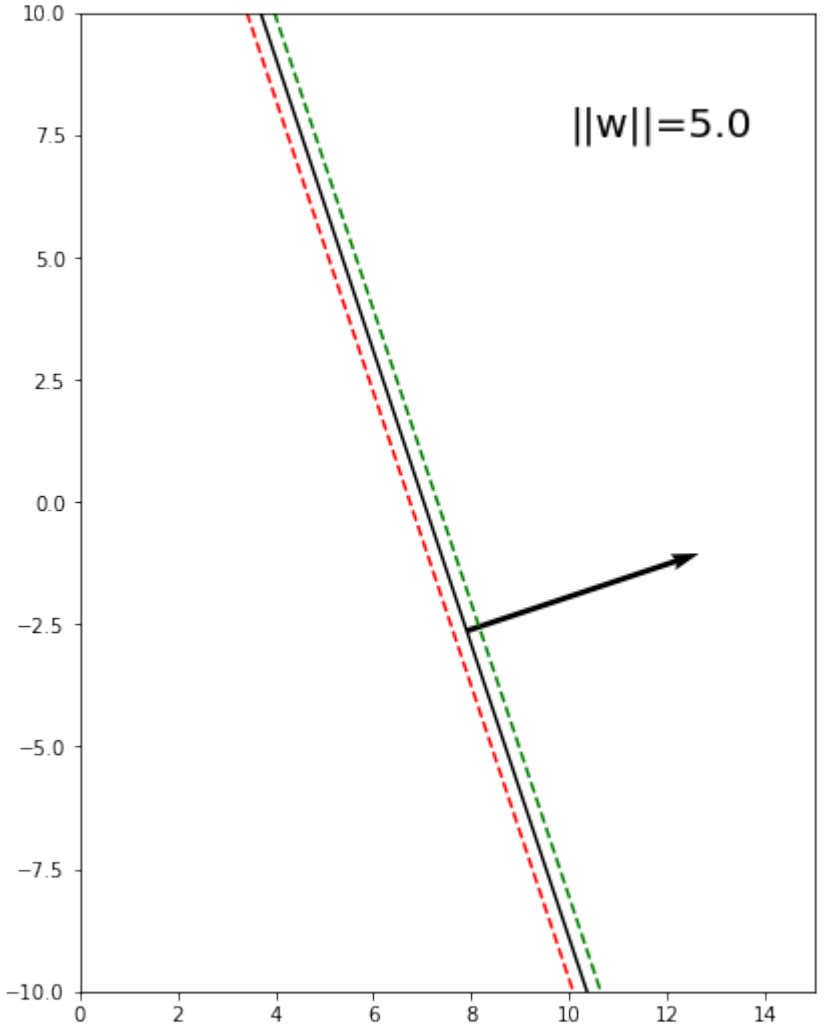
- Makes a space around the classifier where points aren't allowed to be – between the green and red lines.
- **w** can be rescaled arbitrarily to make this true without changing the classifier's expressiveness.

Step 3) Set Up A Constrained Optimization Problem

w can be rescaled to make this true without changing the classifier's expressiveness.







Step 3) Set Up A Constrained Optimization Problem

How does this change our "ugly" objective to something nicer?

$$\max_{w,b} \min_{i} \frac{|w^T x_i + b|}{\|w\|} \quad s.t.\ y_i(w^T x_i + b) > 0 \quad \forall i$$

$$\Rightarrow$$

$$\max_{w,b} \frac{1}{\|w\|} \quad s.t.\ y_i(w^T x_i + b) \geq 1 \quad \forall i$$

You: Neat math. Cute fish. How does this relate to the margin?

Question Break!

For a dataset $\{x_i, y_i\}_{i=1}^n$, how many constraints do we have?

$$\max_{\boldsymbol{w},b} \frac{\mathbf{1}}{\|\boldsymbol{w}\|}$$

$$s.t.\ y_i(w^T x_i + b) \geq 1 \quad \forall i$$

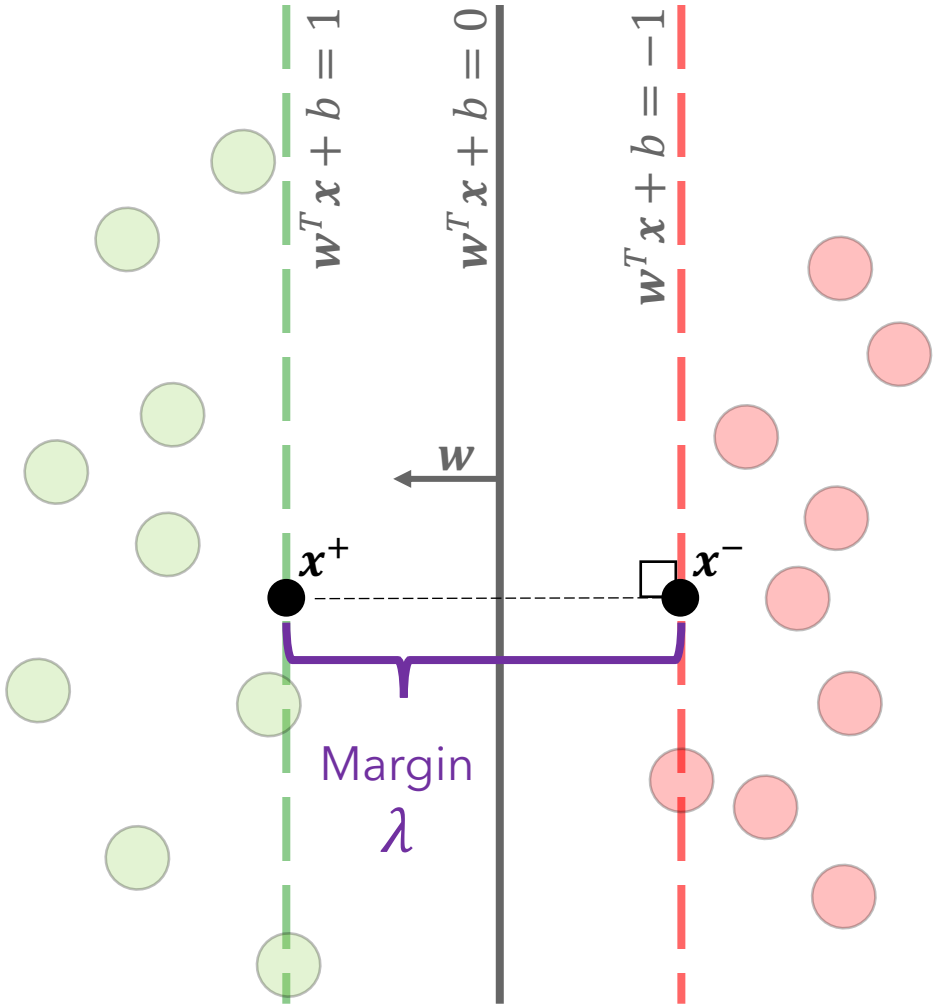
A 1

B d

C n

D 2

Step 2) Derive and Expression for the Margin (Back Again)



Things we assumed plus some geometry:

1) $\boldsymbol{w^T x^- + b = -1}$

2) $\boldsymbol{w^T x^+ + b = 1}$

3) $\boldsymbol{x^+ - x^- = \lambda w}$

Goal: Derive an expression for the margin λ

3) $\Rightarrow \boldsymbol{x^+ = \lambda w + x^-}$

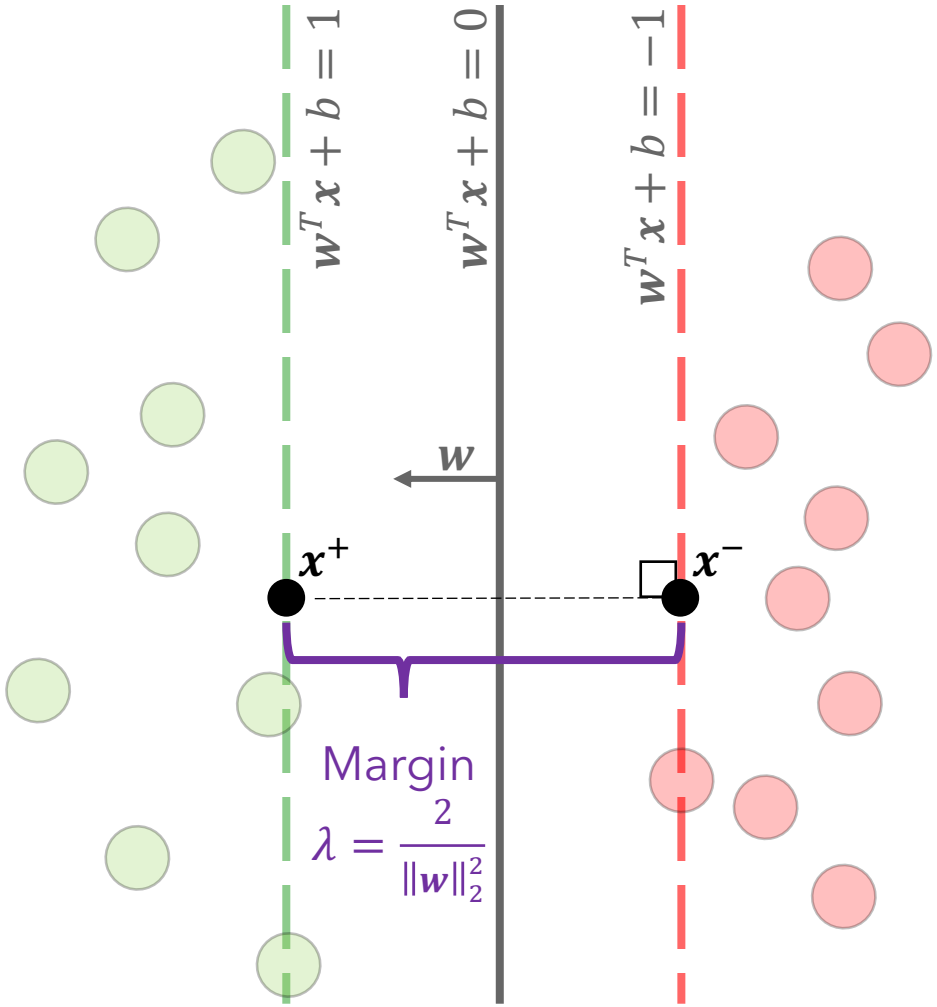
2) $\Rightarrow \boldsymbol{w^T(\lambda w + x^-) + b = 1}$

$\boldsymbol{\lambda w^T w + w^T x^- + b = 1}$

1) $\Rightarrow \boldsymbol{\lambda w^T w - 1 = 1}$

$$\Rightarrow \lambda = \frac{2}{\boldsymbol{w^T w}} = \frac{2}{\|\boldsymbol{w}\|_2^2}$$

Step 3) Set Up A Constrained Optimization Problem (Again)



Maximize margin without making errors:

$$\max_{w,b} \frac{2}{\|w\|_2^2}$$

Maximize margin

$$s.t. \quad y_i(w^T x_i + b) \geq 1 \ \ \forall i$$

Such that no points are misclassified

Recap Steps 1-3: We derived two objectives. How do they relate?

Version 1) Maximize minimum distance to boundary without making errors:

$$\max_{\boldsymbol{w},b} \frac{\mathbf{1}}{\|\boldsymbol{w}\|_2}$$

$$s.t.\ y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \quad \forall i$$

Version 2) Maximize margin width without making errors

$$\max_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|_2^2}$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \ \ \forall i$$

These are proportional! Both are maximized by the same w*.

Could also write this problem as:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \ \ \forall i$$

Hard-Margin Support Vector Machine Objective

How do we actually solve this? And by "solve" I mean optimal find **w** and b

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \;\; \forall i$$

This is a quadratic function with linear constraints. Many (non-trivial) algorithms to solve these sorts of things – called Quadratic Program Solvers (QP-Solvers).

Hard-Margin Support Vector Machine Objective

What is a Quadratic Program Solver? Given equations of the form:

$$\min_{z} \frac{1}{2} z^T P z + q^T z$$

Quadratic equation in matrix notation

$$s.t. \quad Gz \leq h$$

Linear inequality constraints

$$Az = b$$

Linear equality constraints

$$b_{lower} \leq z \leq b_{upper}$$

Limits on the values of x

Runs algorithms from optimization to provide the vector z^* that minimizes the function without violating the constraints. (Or returns infeasible if no such solution exits)

- One example from Python (https://pypi.org/project/qpsolvers/)

Hard-Margin Support Vector Machine Objective

How can we put this into a quadratic form?

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \ \ \forall i$$

$$\min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^T P\boldsymbol{z} + \boldsymbol{q}^T\boldsymbol{z}$$

$$s.t. \quad G\boldsymbol{z} \leq \boldsymbol{h}$$
$$A\boldsymbol{z} = \boldsymbol{b}$$
$$\boldsymbol{b_{lower}} \leq \boldsymbol{z} \leq \boldsymbol{b_{upper}}$$

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}_{\boldsymbol{d+1\times 1}}$$

$$P = \begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix}_{d+1\times d+1}$$

$$\boldsymbol{q} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}_{\boldsymbol{d+1}}$$

$$\boldsymbol{h} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{\boldsymbol{d+1}}$$

$$G = \begin{bmatrix} y_1\boldsymbol{x_1^T}, y_1 \\ y_2\boldsymbol{x_2^T}, y_2 \\ y_3\boldsymbol{x_3^T}, y_3 \\ \vdots \\ y_n\boldsymbol{x_n^T}, y_n \end{bmatrix}_{n\times d+1}$$

Hard-Margin Support Vector Machine Objective

How can we put this into a quadratic form?

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

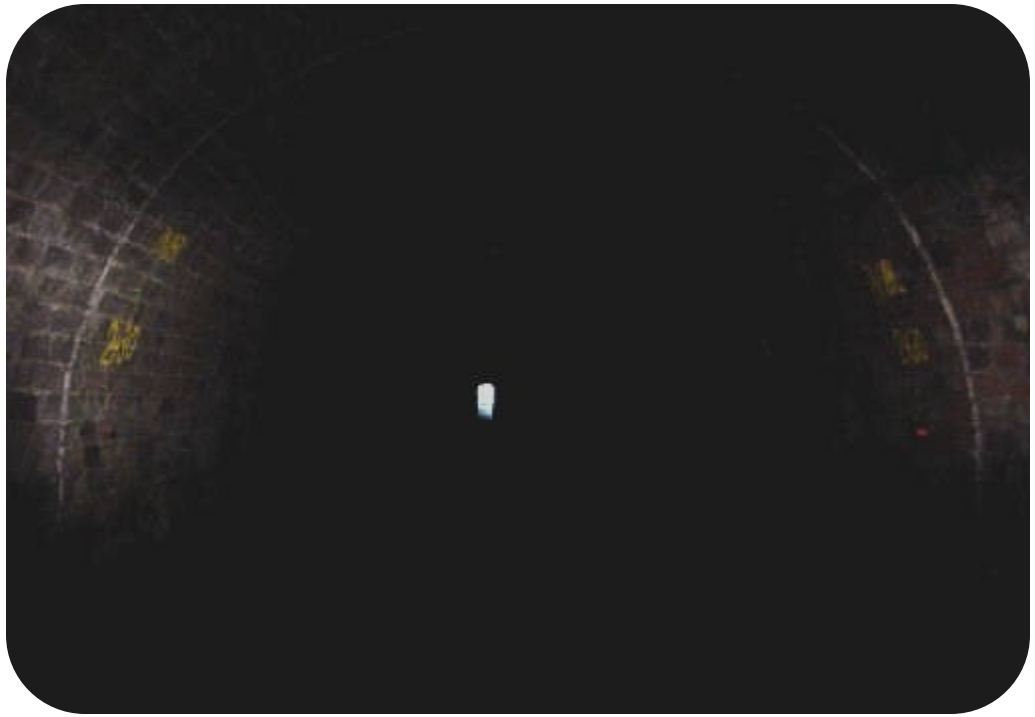
$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \;\; \forall i$$

$$\min_{\boldsymbol{w},b} \frac{1}{2}[\boldsymbol{w}\; b]\begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} + \vec{0}^T\begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}$$

$$s.t. \begin{bmatrix} y_1\boldsymbol{x_1^T}, y_1 \\ y_2\boldsymbol{x_2^T}, y_2 \\ y_3\boldsymbol{x_3^T}, y_3 \\ \vdots \\ y_n\boldsymbol{x_n^T}, y_n \end{bmatrix}\begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} \geq \vec{\mathbf{1}}$$

Can pass this to a QP solver and let some optimization people deal with it.

For our purposes however, there is some deeper bit of understanding to be gained from pushing this further with another method – Lagrange multipliers.



Constrained Optimization with Lagrange Multipliers

To be *very* clear, the following is beyond the scope of this class and would be content in an optimization course. But we need the tool to move forward with our work here.

I'm going through this for completeness and to give you intuition for when I apply it later.

If you understand it, great. If not, don't worry and focus on the outcome and conclusions.

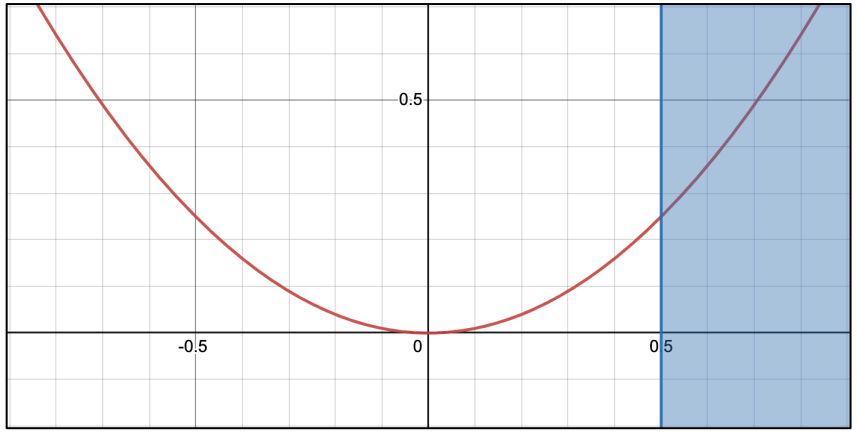
Constrained Optimization with Lagrange Multipliers

Suppose we need to minimize some function given a set of constraints:

$$\min_x f(x)$$
$$s.t.\ g_i(x) \leq 0 \ \text{ for } i = 1,2, \ldots m$$

For example, a quadratic function with one linear constraint:

$$\min_x \ x^2$$
$$s.t.\ -10x + 5 \leq 0$$



In general, f and g_i can be arbitrary.

Constrained Optimization with Lagrange Multipliers

Write down an unconstrained function called the Lagrangian:

$$\mathcal{L}(x, \alpha_1, \ldots, \alpha_m) = f(x) + \sum_i^m \alpha_i g_i(x)$$

The original optimization problem is equivalent to solving the following:

Primal: $\min_x \max_{\alpha_1,\ldots,\alpha_m} \mathcal{L}(x, \alpha_1, \ldots, \alpha_m) \quad s.t.\ \alpha_i \geq 0\ \forall i$

By exchanging the order of min and max, we get the dual problem:

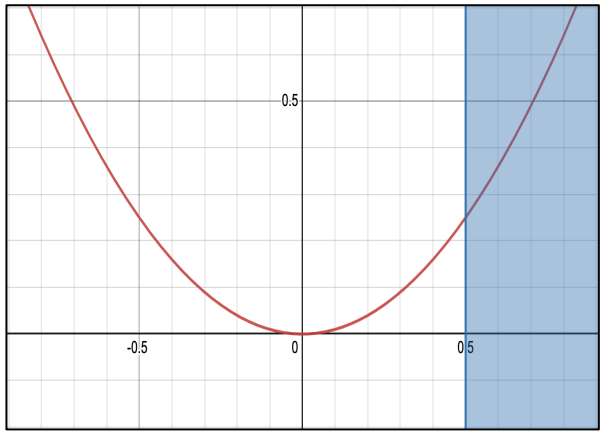
Dual: $\max_{\alpha_1,\ldots,\alpha_m} \min_x \mathcal{L}(x, \alpha_1, \ldots, \alpha_m) \quad s.t.\ \alpha_i \geq 0\ \forall i$

Constrained Optimization with Lagrange Multipliers

$$\min_x \; x^2 \quad s.t. \; -10x + 5 \leq 0$$

➡ $$\mathcal{L}(x, \alpha) = x^2 + \alpha(-10x + 5)$$

Dual: $$\max_{\alpha} \min_{x} \; x^2 + \alpha(-10x + 5) \qquad s.t. \; \alpha_i \geq 0 \; \forall i$$



$$\frac{d\mathcal{L}(x, \alpha)}{dx} = 2x - 10\alpha = 0 \quad \Rightarrow \quad x = 5\alpha$$

$$\frac{d\mathcal{L}(x, \alpha)}{d\alpha} = -10x + 5 = 0$$

$$\Rightarrow -50\alpha + 5 = 0$$

$$\Rightarrow \alpha = \frac{1}{10} \qquad \Rightarrow x = \frac{1}{2}$$

Constrained Optimization with Lagrange Multipliers

$$\mathcal{L}(x, \alpha_1, \ldots, \alpha_m) = f(x) + \sum_i^m \alpha_i g_i(x)$$

Think of these like a 2-player adversarial games with different turn order.

Primal: $\min_x \max_{\alpha_1,\ldots,\alpha_m} \mathcal{L}(x, \alpha_1, \ldots, \alpha_m) \quad s.t.\ \alpha_i \geq 0\ \forall i$

Dual: $\max_{\alpha_1,\ldots,\alpha_m} \min_x \mathcal{L}(x, \alpha_1, \ldots, \alpha_m) \quad s.t.\ \alpha_i \geq 0\ \forall i$

In the **primal**, the x-player goes first and tries to set whatever value of x minimizes things (ignoring constraints). Then the α-player goes and can set huge penalties anywhere a constraint is violated.

In the **dual**, the α-player goes first and tries to maximize things. Then the x-player goes and can try to minimize from there.

Constrained Optimization with Lagrange Multipliers

$$\mathcal{L}(x, \alpha_1, \ldots, \alpha_m) = f(x) + \sum_i^m \alpha_i g_i(x)$$

Duality Gap: max of min is often lower than min of max.

Primal: $\min_x \max_{\alpha_1,\ldots,\alpha_m} \mathcal{L}(x, \alpha_1, \ldots, \alpha_m) \quad s.t.\ \alpha_i \geq 0\ \forall i$

Dual: $\max_{\alpha_1,\ldots,\alpha_m} \min_x \mathcal{L}(x, \alpha_1, \ldots, \alpha_m) \quad s.t.\ \alpha_i \geq 0\ \forall i$

Karush-Kuhn-Tucker (KKT) Conditions tell us when the dual and primal have the same optimal solutions $x^*, \alpha_1^*, \ldots, \alpha_m^*$ for convex $f(x)$:

- $\nabla\mathcal{L}(x^*, \alpha_1^*, \ldots, \alpha_m^*) = 0$ — At a critical point of the Lagrangian
- $g(x^*) \leq 0$ — The solution satisfies the primal constraints
- $\alpha_i^* \geq 0$ — The solution satisfies the dual constraints
- $\alpha^* g(x^*) = 0$ — Complementary slackness.

Constrained Optimization with Lagrange Multipliers

$$\min_x \; x^2$$
$$s.t. \; -10x + 5 \leq 0$$

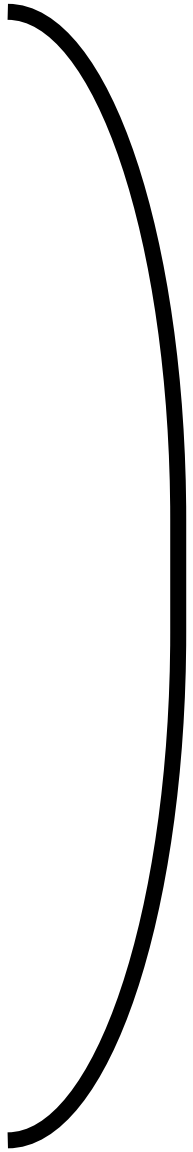
$$\mathcal{L}(x, \alpha) = x^2 + \alpha(-10x + 5)$$

$$x^* = \frac{1}{2} \qquad \alpha^* = \frac{1}{5}$$

Karush-Kuhn-Tucker (KKT) Conditions tell us when the dual and primal have the same optimal solutions $x^*, \alpha_1^*, \ldots, \alpha_m^*$ for convex $f(x)$:

- $\frac{\delta \mathcal{L}(x^*, \alpha^*)}{dx} = 0, \frac{\delta \mathcal{L}(x^*, \alpha^*)}{d\alpha} = 0$ — At a critical point of the Lagrangian
- $-10x^* + 5 \leq 0$ — The solution satisfies the primal constraints
- $\alpha^* \geq 0$ — The solution satisfies the dual constraints
- $\alpha^*(-10x^* + 5) = 0$ — Complementary slackness.

Constrained Optimization with Lagrange Multipliers



Hard-Margin Support Vector Machine Objective

Where were we? We wanted to solve the constrained optimization problem below because it maximizes the margin while having no errors.

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \;\; \forall i$$

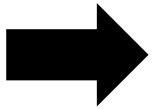
Let's start getting this into shape to apply the Langrange multipliers bit.

Hard-Margin Support Vector Machine Objective

Need to get our inequality facing the right way. Multiply by -1 and move the right side over:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \;\; \forall i$$



$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t. \quad 1 - y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \leq 0 \;\; \forall i$$

Hard-Margin Support Vector Machine Objective

$$\min_{w,b} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$
$$s.t. \ \ 1 - y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \leq 0 \ \ \forall i$$

Let's write out the Lagrangian:

$$\mathcal{L}(w, b, \alpha_i, \dots, \alpha_n) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b)\right)$$

$$= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i\, y_i\, \boldsymbol{w}^T\boldsymbol{x_i} - \sum_{i=1}^{n} \alpha_i\, y_i\, b$$

Want to solve the dual problem:

$$\max_{\alpha_i \geq 0} \min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i\, y_i\, \boldsymbol{w}^T\boldsymbol{x_i} - \sum_{i=1}^{n} \alpha_i\, y_i\, b$$

Hard-Margin Support Vector Machine Objective

$$\max_{\alpha_i \geq 0} \min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i\, y_i\, \boldsymbol{w}^T\boldsymbol{x_i} - \sum_{i=1}^{n} \alpha_i\, y_i\, b$$

Take derivatives and set equal to zero:

$$\frac{\delta \mathcal{L}}{\delta \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i\, y_i\, \boldsymbol{x_i} = \boldsymbol{0} \quad \Rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i\, y_i \boldsymbol{x_i}$$

$$\frac{\delta \mathcal{L}}{\delta b} = -\sum_{i=1}^{n} \alpha_i\, y_i = 0$$

Hard-Margin Support Vector Machine Objective

$$\max_{\alpha_i \geq 0} \min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\, y_i\, \boldsymbol{w}^T\boldsymbol{x_i} - \sum_{i=1}^{n}\alpha_i\, y_i\, b$$

$$\boldsymbol{w} = \sum_{i=1}^{n}\alpha_i\, y_i \boldsymbol{x_i} \qquad \sum_{i=1}^{n}\alpha_i\, y_i = 0$$

Apply what we learned from the derivatives:

$$\max_{\alpha_i \geq 0} \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i\, y_i \boldsymbol{x_i}\right)^T\left(\sum_{j=1}^{n}\alpha_j\, y_j \boldsymbol{x_j}\right) + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\, y_i\left(\sum_{j=1}^{n}\alpha_j\, y_j \boldsymbol{x_j}\right)^T \boldsymbol{x_i} - b\sum_{i=1}^{n}\alpha_i\, y_i$$

$$\max_{\alpha_i \geq 0} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x_i^T x_j} \qquad s.t. \sum_{i=1}^{n}\alpha_i\, y_i = 0$$

Hard-Margin Support Vector Machine Objective

$$\max_{\alpha_i \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i}^{n} \sum_{j}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^T x_j} \qquad s.t. \sum_{i=1}^{n} \alpha_i \, y_i = 0$$

This is a quadratic function with linear constraints. Many (non-trivial) algorithms to solve these sorts of things – called Quadratic Program Solvers (QP-Solvers). I can put this into a QP solver and get back out optimal α_i^*.

Further, **the KKT conditions hold** (not shown here) such that solving this yields the exact same optimal solution as the original formalization!

Please accept this majestic photo of my friend's cat Simba for sticking with me

Recap of what we've been doing so far

Motivation: We wanted to find a classifier that maximizes the margin between classes.

Derivation: We derived the following constrained optimization problem that does that:

Hard-Margin SVM Primal:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \qquad s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \;\; \forall i$$

We said that is a Quadratic Programming problem and totally solvable. Then I ran us through a lot of math to arrive at the dual formalization below which solves the same problem (KKT's hold).

Hard-Margin SVM Dual:

$$\max_{\alpha_i \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x_i^T x_j} \qquad s.t. \sum_{i=1}^{n} \alpha_i \, y_i = 0$$

Which is also a Quadratic Programming problem (in α's this time rather than **w**) and is totally solvable to global optimality….. **…. so why did I do this to us all?**

Why have we done all this?

Hard math builds character.

Examining the Dual Solution

Real Reason: We can examine this dual formalization to observe some very, very interesting things that are not at all apparent in the primal formalization!

Observation 1: For optimal α_i^* found by maximizing the dual problem, the optimal weight vector turns out to be a weighted sum of input vectors times their labels:

We found this back when we took derivatives.

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* \, y_i \boldsymbol{x_i}$$

Arises from KKT conditions, next slide.

$$b^* = \frac{1}{\sum_{i=1}^{n} \alpha_i^*} \sum_{i=1}^{n} \alpha_i^* (y_i - \boldsymbol{w}^{*T} \boldsymbol{x_i})$$

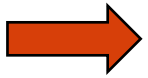
Examining the Dual Solution

Observation 2: Only certain training examples influence the result. Many of the α_i's are zero.

At optimality, the KKT conditions require that:

$$\alpha_i \geq 0 \; \forall i$$ ← Satisfies dual constraints

$$y_i\left(w^{*T} x_i + b\right) \geq 1, \qquad \forall i$$ ← Satisfies primal constraints



$$\alpha_i^*\left(y_i\left(w^{*T} x_i + b\right) - 1\right) = 0, \qquad \forall i$$ ← Complementary slackness

Questions Break!

What does the complementary slackness condition

$\alpha_i^*\left(y_i\left(w^{*T}x_i + b\right) - 1\right) = 0,\ \forall i$ imply about the values of the alpha variables?

(Think about when this equation is true for a single training point i)

Examining the Dual Solution

Observation 2: Only certain training examples influence the result. Many of the α_i's are zero.

At optimality, the KKT conditions require that:

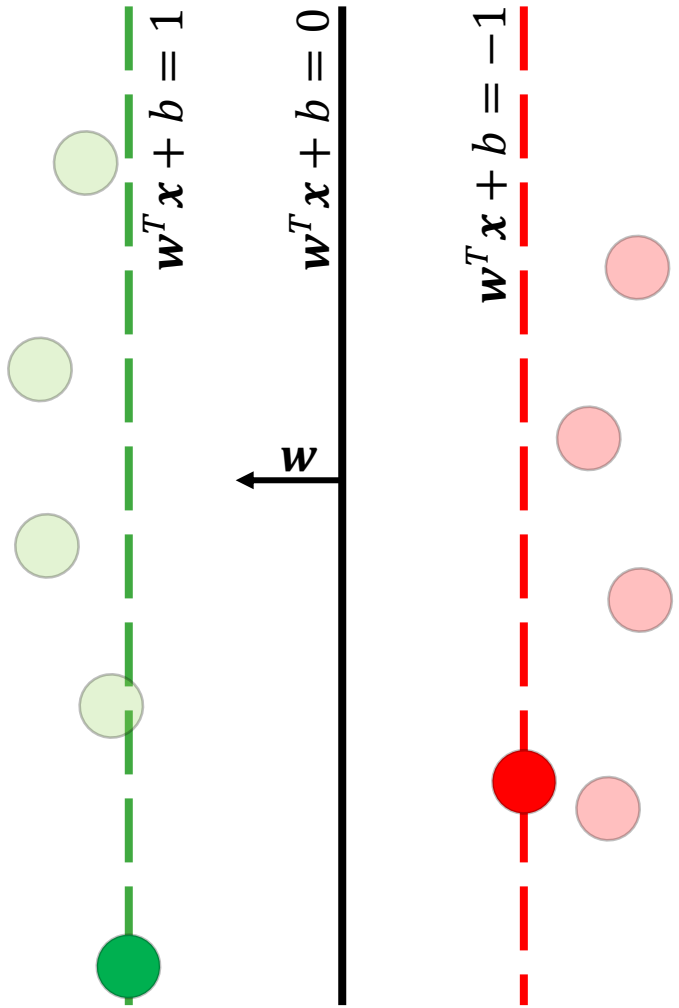
$$\alpha_i \geq 0 \; \forall i$$ ← Satisfies dual constraints

$$y_i\left(w^{*T} x_i + b\right) \geq 1, \qquad \forall i$$ ← Satisfies primal constraints

$$\alpha_i^*\left(y_i\left(w^{*T} x_i + b\right) - 1\right) = 0, \qquad \forall i$$ ← Complementary slackness

α_i^* is only non-zero if $y_i\left(w^T x_i + b\right) = 1$!

Examining the Dual Solution



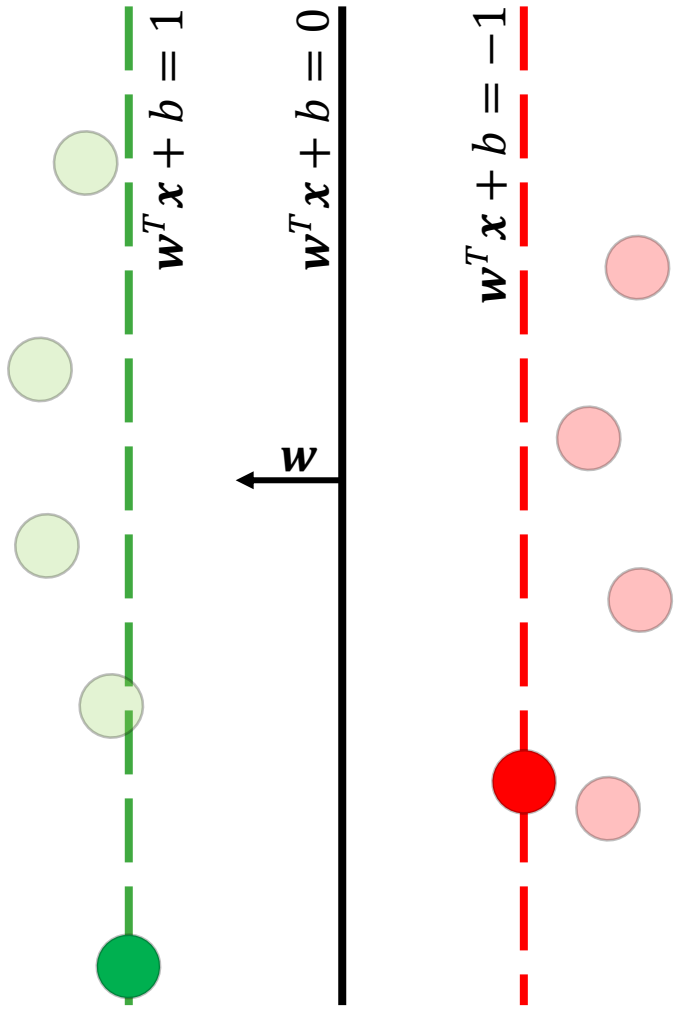
α_i^* **is only non-zero if** $y_i(w^T x_i + b) = 1$**!**

α_i^* is only non-zero if x_i is on the margin edges (green or red line)

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$$

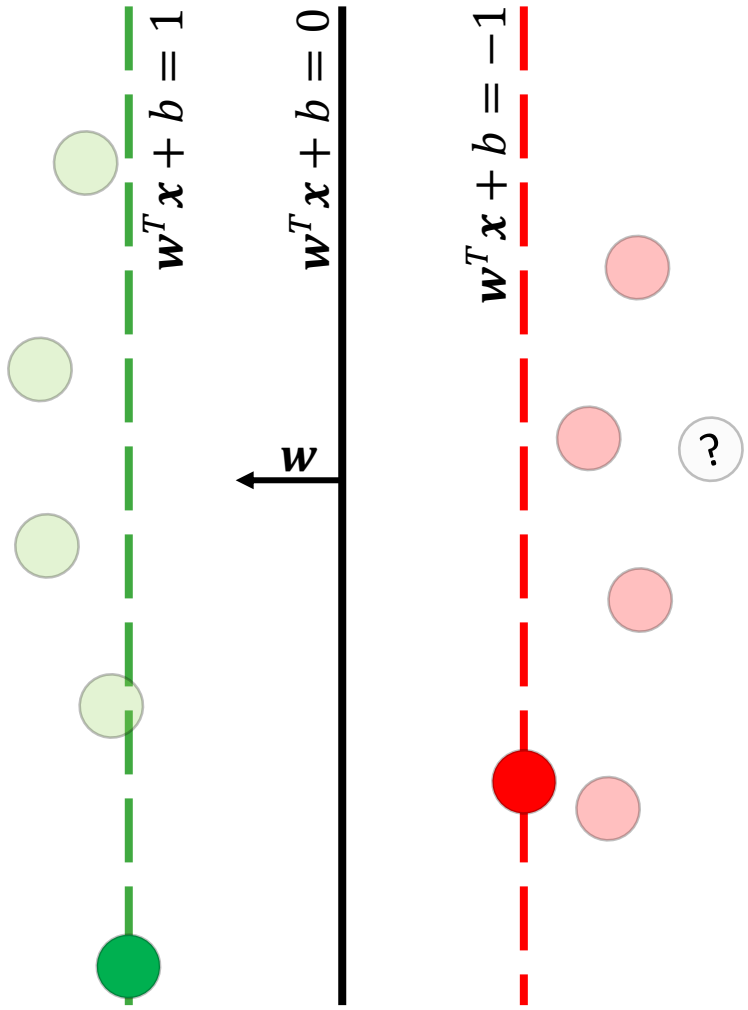
Only the points on the margin matter to the classifier weights.

Examining the Dual Solution



Call these points with non-zero alpha **support vectors.**

Examining the Dual Solution



How is a new point classified?

Our decision rule is:

$$\boldsymbol{w}^T\boldsymbol{x}+b>0 \;\Rightarrow\; predict\ 1$$
$$\boldsymbol{w}^T\boldsymbol{x}+b<0 \;\Rightarrow\; predict\ -1$$

Plugging in our optimal weight:

$$\boldsymbol{w}^*=\sum_{i=1}^{n}\alpha_i^*\,y_i\boldsymbol{x_i} \quad\Rightarrow\quad \boldsymbol{w^T x}+b=b+\sum_{i=1}^{n}\alpha_i^*\,y_i\boldsymbol{x_i^T x}$$

The prediction is based on dot product with a sparse set of training examples.

Web Demo

https://jgreitemann.github.io/svm-demo

What you need to know from today's lecture:

Support Vector Machines (SVMs) find the decision boundary that maximizes the margin between the positive and negative points.

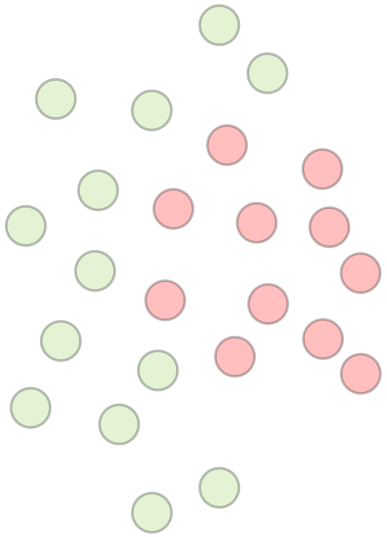
The weight vector corresponding to this max-margin classifier can be found by solving a constrained optimization problem with a QP solver.

By going through the effort of examining the Lagrangian dual of the SVM objective, we found that the **decision boundary is a function of a small, sparse set of examples** that lie on the classifier's margin.

What we derived today is referred to as a **hard-margin SVM** because it does not tolerate *any* errors. If we give it a non-linearly separable dataset, the QP solver will report no feasible solution.

Teaser for Next Time: What about the non-linearly separable case?

Teaser 1: We can modify our objective function to try to maximize margin while making as few mistakes as possible. Will need to introduce some notion of the tradeoff between the two.



Maximize margin

Minimize violations of the margin

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \epsilon_i$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 - \epsilon_i \;\; \forall i$$
$$\epsilon_i \geq 0$$

Allow some violations

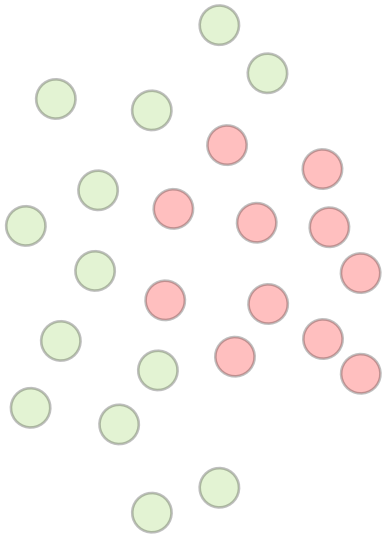
We'll talk about how this works out next time.

Teaser for Next Time: What about the non-linearly separable case?

Teaser 2: Apply the non-linear basis function idea again and just replace each point with some $\Phi(x)$ that is non-linear. Then fit a linear classifier in that higher-dimensional space.

Train and find $\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* \, y_i \Phi(\boldsymbol{x_i})$

Predict based on $\boldsymbol{w^T}\Phi(\boldsymbol{x}) + b$ ⟶ $b + \sum_{i=1}^{n} \alpha_i^* \, y_i \boldsymbol{\Phi}(\boldsymbol{x_i})^{\boldsymbol{T}} \boldsymbol{\Phi}(\mathbf{x})$



This is expensive and we have to deal with those higher dimensions. What if we just defined a kernel that compute non-linear similarity without actually representing these high dimensional vectors.

$$b + \sum_{i=1}^{n} \alpha_i^* \, y_i \boldsymbol{\mathcal{K}}(\mathbf{x_i}, \boldsymbol{x})$$

$$\max_{\alpha_i \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i}^{n} \sum_{j}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{\mathcal{K}}(\mathbf{x_i}, \boldsymbol{x_j})$$

SVMs operate on dot products between examples. Can "pretend" we are in a higher dimension – call this the **kernel trick**. More next time.

Next time

Next Time: How do we deal with non-linearly separable data? Kernels + Soft-Margin SVM.