University of Colorado **Boulder**

Department of Computer Science

CSCI 5622: Machine Learning

Chris Ketelsen

Lecture 12: Support Vector Machines
Hard-Margin SVM

Binary Classification

Given: $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ training examples

$$\mathbf{x}_i \in \mathbb{R}^D \qquad y_i \in \{c_1, c_2\}$$

Goal: Given new data $\mathbf{x}$, predict its label y

Already Seen:

- Naive Bayes
- Logistic Regression

Starting Today:

- Support Vector Machines (SVMs)

Support Vector Machines

Advantages:

- Allows nonlinear classification
- Surprisingly nice theoretical bounds
- Optimization problem for weights is convex
- "Best off-the-shelf classifier" - Andrew Ng

Disadvantages:

- Can be prone to overfitting
- No firm probabilistic interpretation

Our Roadmap

Today:

- Talk linear SVM when data is linearly separable

Next Time:

- Talk linear SVM when data is not linearly separable
- Look at efficient algorithm for finding weights

Next Next Time:

- Look at SVM as a nonlinear classifier
- Learn the Kernel Trick

Linear Classifiers

All of these are examples of **linear classifiers**

We'll introduce SVMs as linear classifiers as well

Then we'll extend SVMs to the nonlinear case

Slight Notational Change:

- Assume binary classes are $y \in \{-1, 1\}$
- Assume parameters are $\mathbf{w} \in \mathbb{R}^D$ and b

Before we had $\mathbf{w}, \mathbf{x}_i \in \mathbb{R}^{D+1}$ and wrote $\mathbf{w}^T \mathbf{x}$

Now we have $\mathbf{w}, \mathbf{x}_i \in \mathbb{R}^D$ and write $\mathbf{w}^T \mathbf{x} + b$

Linear Classifiers

def: A linear classifier computes a weighted sum (via a dot product) of the features of the form $\mathbf{w}^T\mathbf{x} + b$ and then classifies based

$$\mathbf{w}^T\mathbf{x} + b \geq 0$$

The boundary of the two classes is defined by $\mathbf{w}^T\mathbf{x} + b$ and is called the **separator** or the **decision boundary**

We estimate the weights (and thus the decision boundary) using the training set

Linear Classifiers

In 1D we have just one feature, x_1

Decision rule $\mathbf{w}^T\mathbf{x} + b \geq 0$ becomes $w_1 x_1 + b \geq 0$, or ...

$$x_1 = -\frac{b}{w_1} \geq 0$$

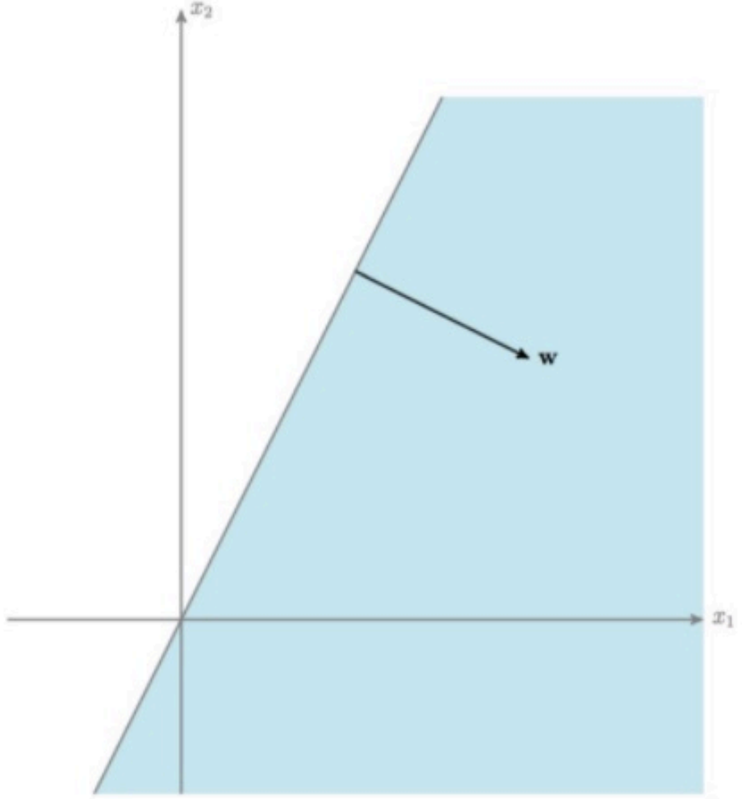
Decision boundary is just the point $-\frac{b}{w_1}$

Linear Classifiers

In 2D we have two features, x_1 and x_2

Decision rule $\mathbf{w}^T\mathbf{x} + b \geq 0$ becomes $w_1x_1 + w_2x_2 + b \geq 0$

Decision boundary is the line $x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$

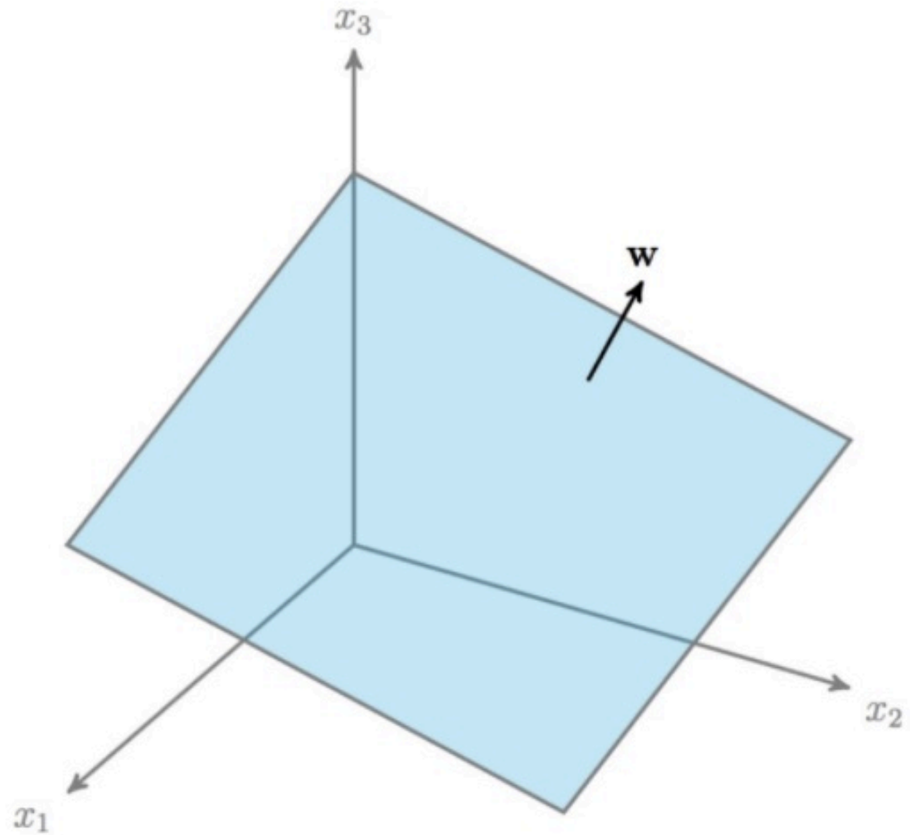


Note that $\mathbf{w}$ is perpendicular (orthogonal) to decision boundary

Linear Classifiers

In 3D we have three features, x_1 , x_2 , and x_3

Decision rule $\mathbf{w}^T\mathbf{x} + b \geq 0$ gives a plane $\mathbf{w}^T\mathbf{x} + b = 0$



Note that $\mathbf{w}$ is perpendicular (normal) to the plane

Linear Classifiers

In higher dimensions it becomes harder to draw picture

We call the decision boundary the **separating hyperplane**

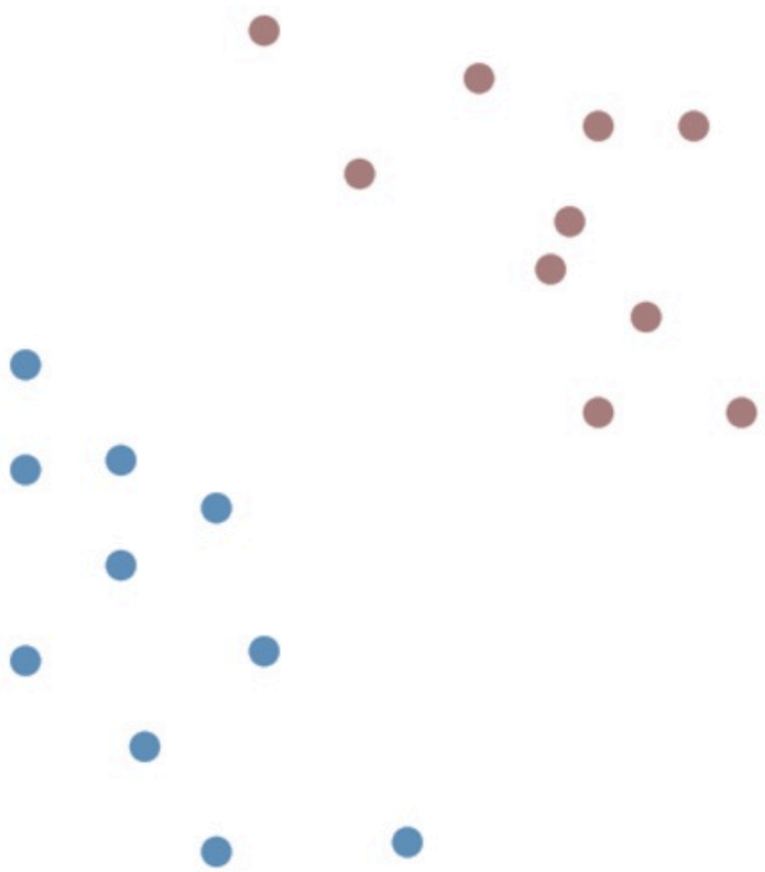
Different types of linear classifiers determine the weights of the separating hyperplane in different ways

- Naive Bayes -- Joint probability and Bayes Rule
- Logistic Regression -- Conditional Likelihood
- SVM -- A more geometric idea: **Margins**

Temporary Assumption: The data is **linearly separable**

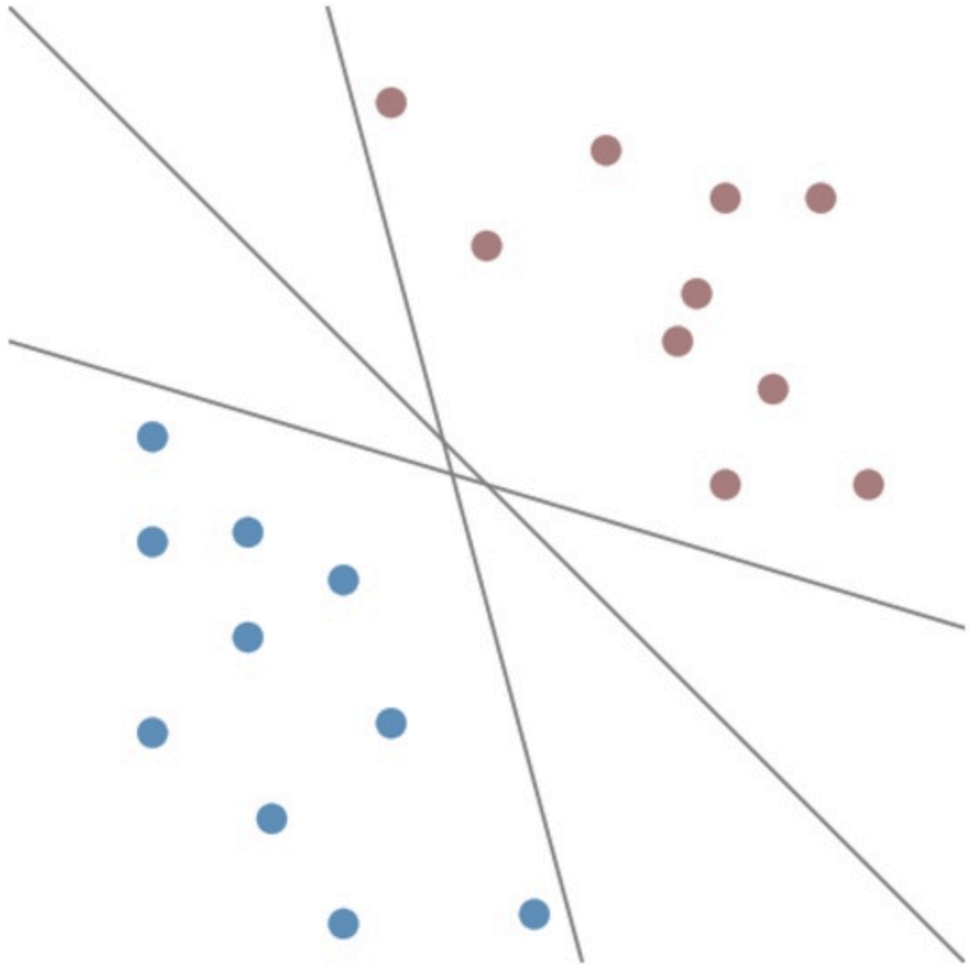
SVM Intuition

Consider the following training data



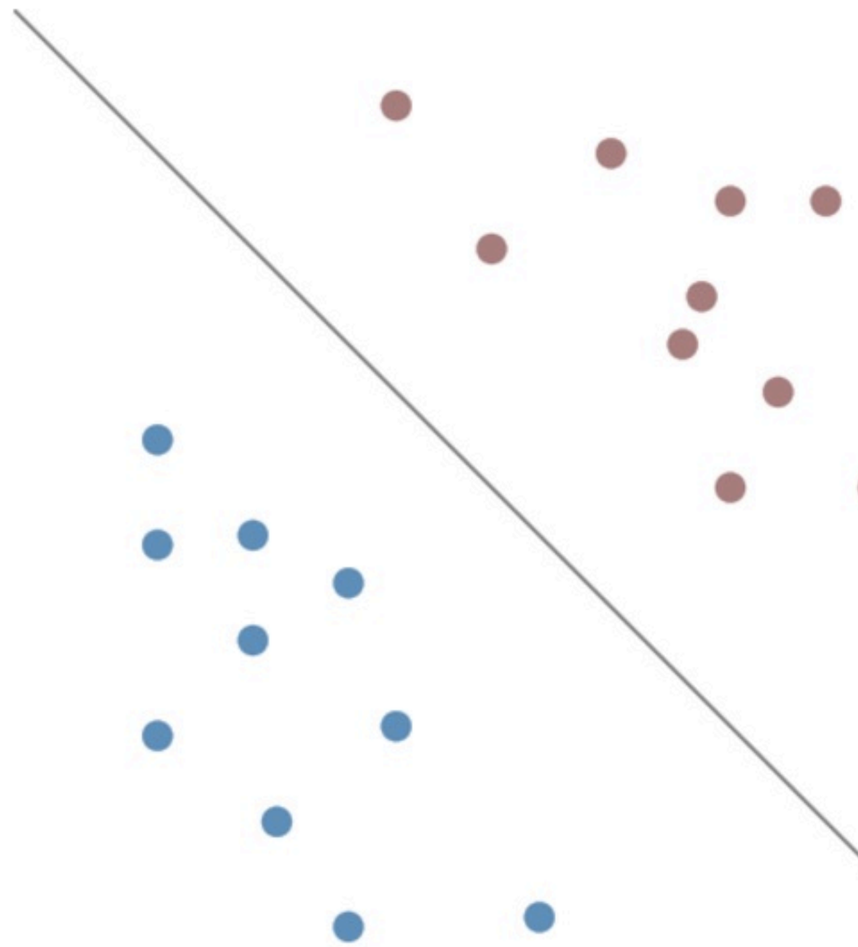
SVM Intuition

Which decision boundary seems **better** to you?



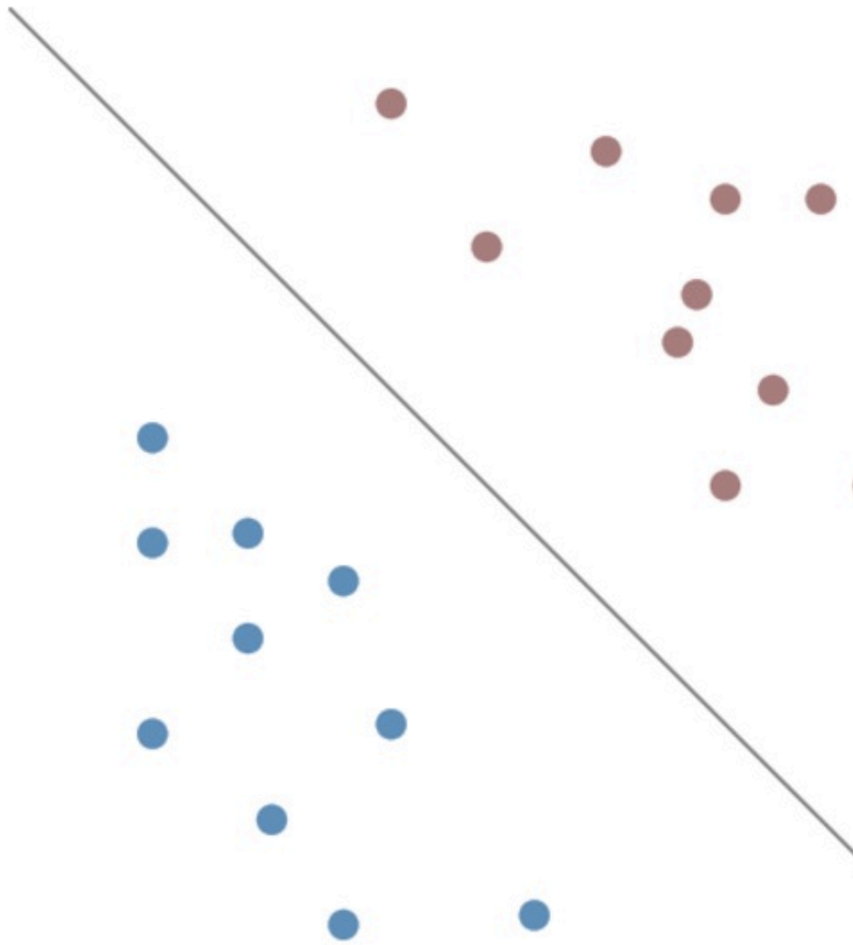
SVM Intuition

This one feels better.



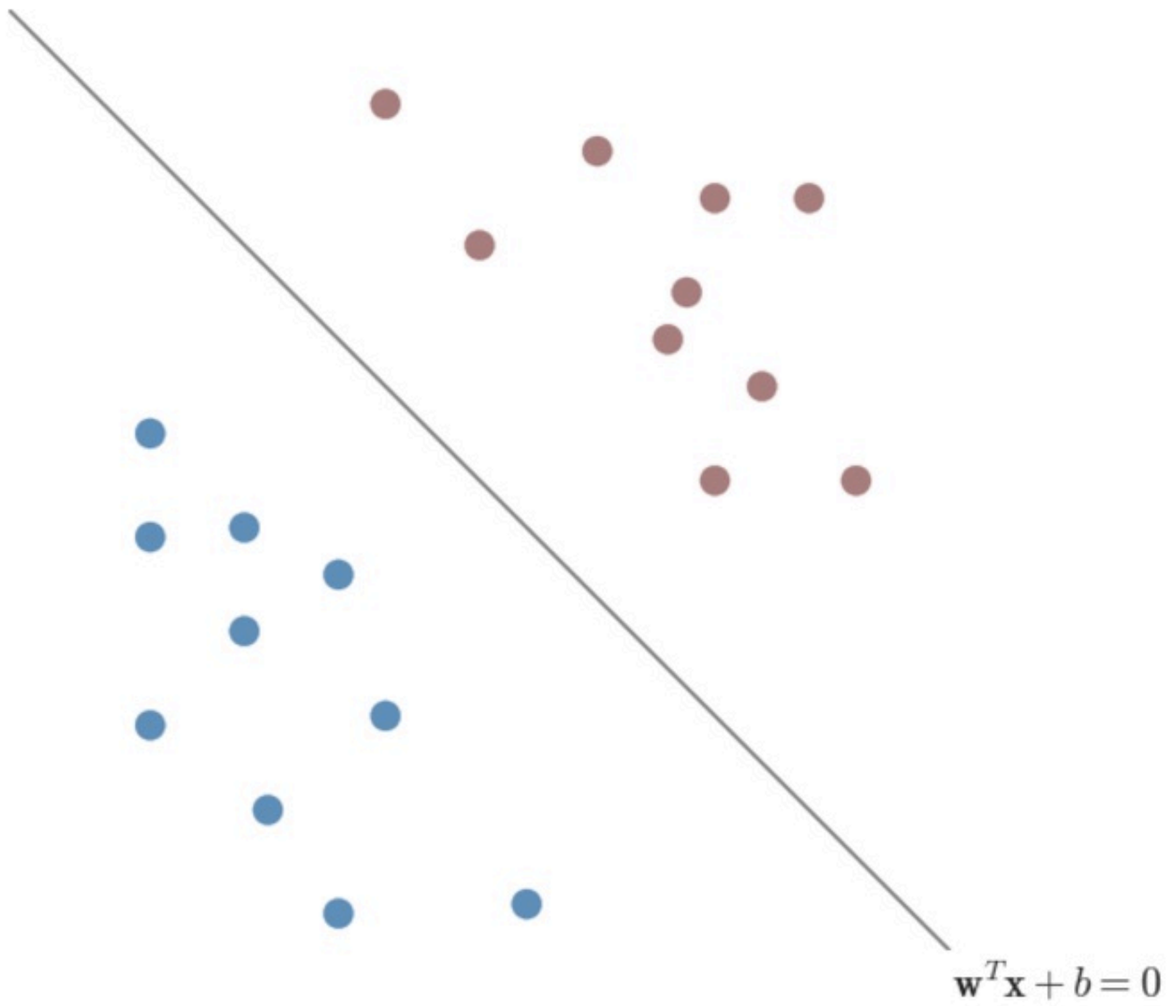
SVM Intuition

More space between DB and closest training points.



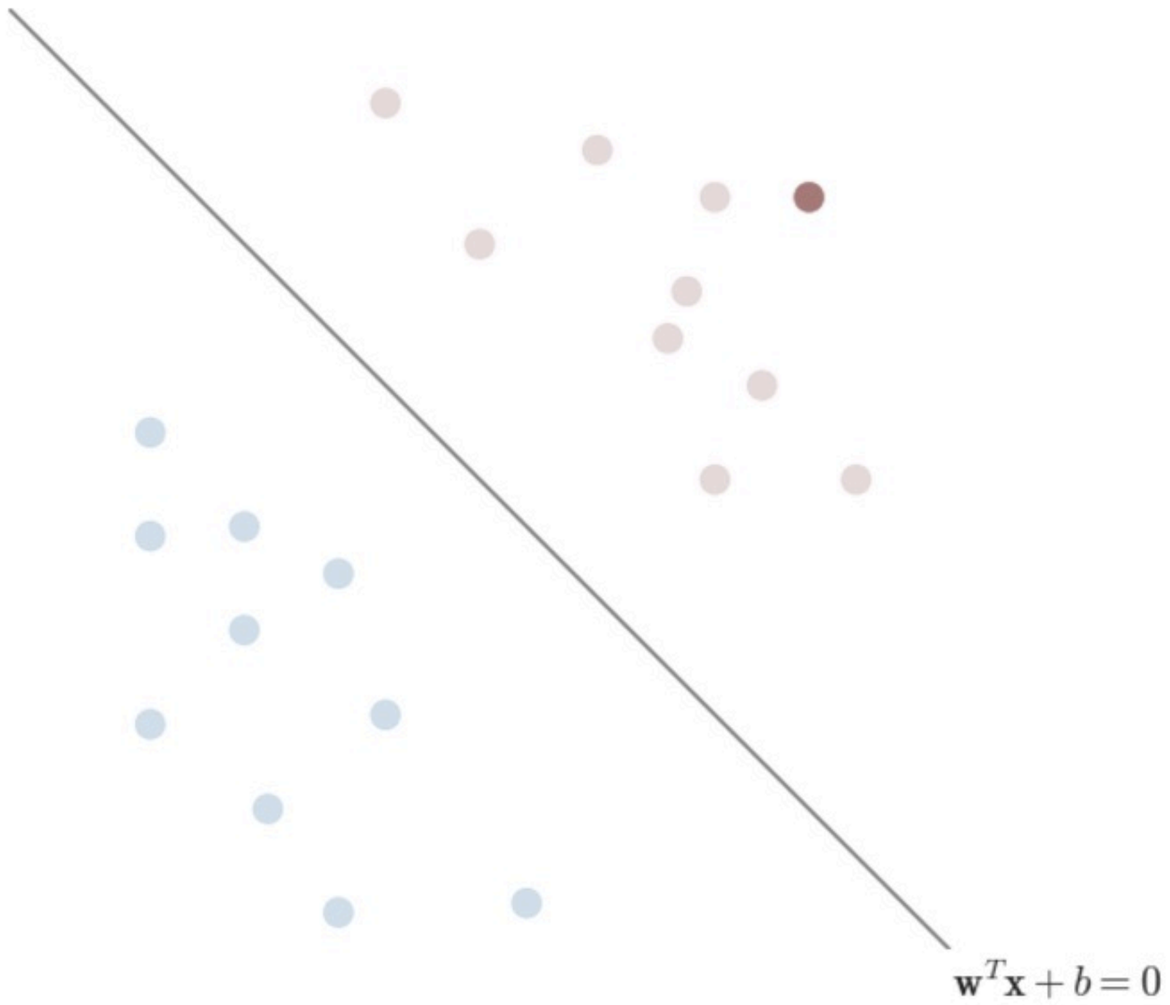
SVM Intuition

Think about size of $\mathbf{w}^T \mathbf{x}_i + b$ as measure of confidence



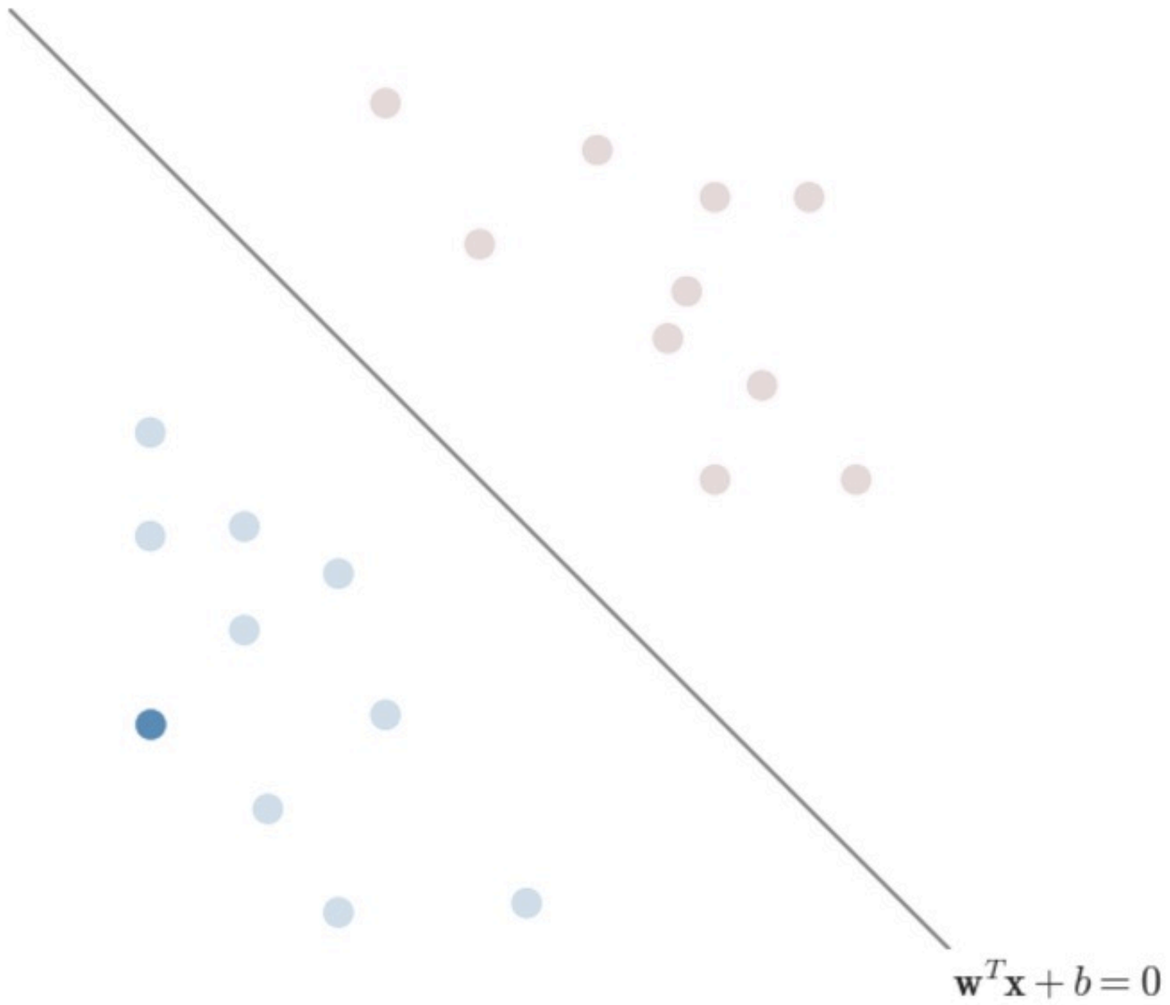
SVM Intuition

Very confident $y_i = 1$ if $\mathbf{w}^T \mathbf{x}_i + b \gg 0$



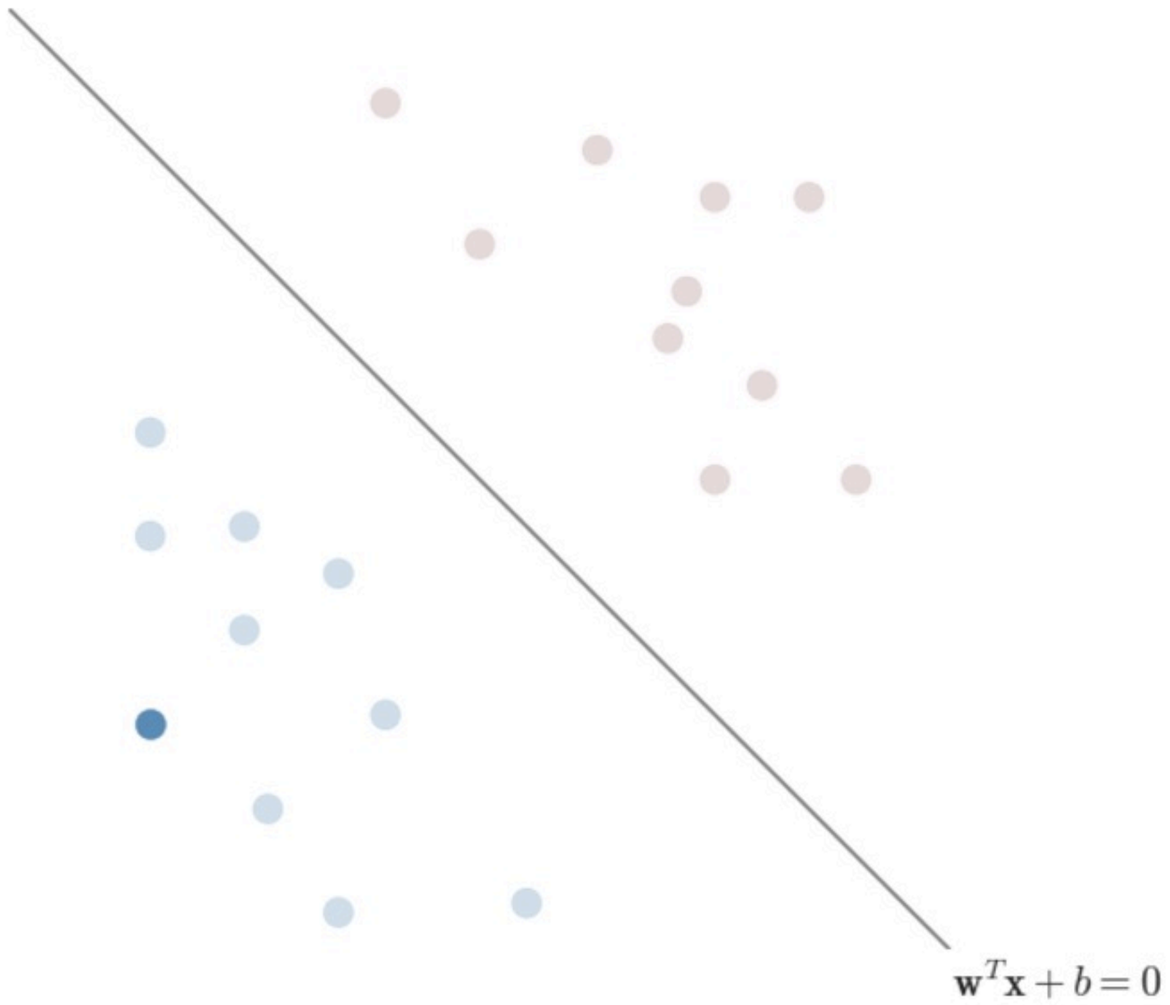
SVM Intuition

Very confident $y_i = -1$ if $\mathbf{w}^T \mathbf{x}_i + b \ll 0$



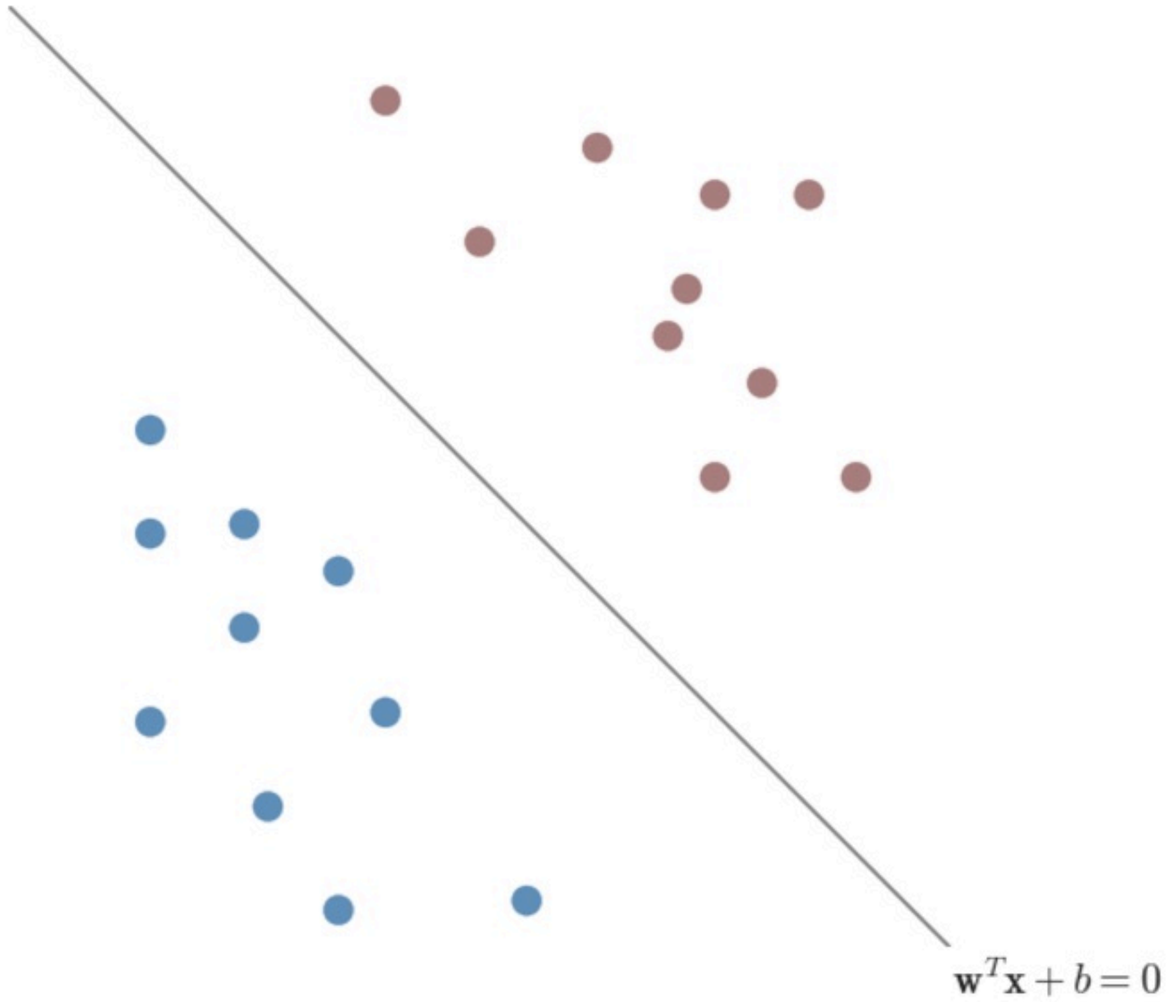
SVM Intuition

Very confident $y_i = -1$ if $\mathbf{w}^T\mathbf{x}_i + b \ll 0$



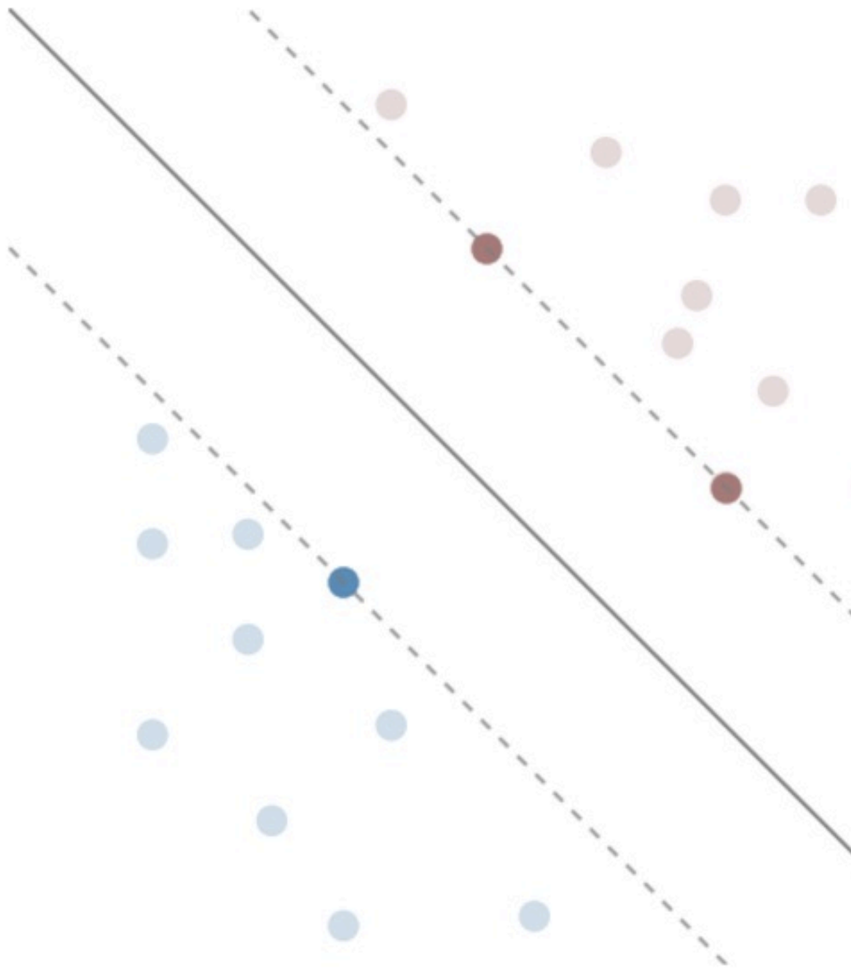
SVM Intuition

Want DB that makes us very confident in classification of each training point



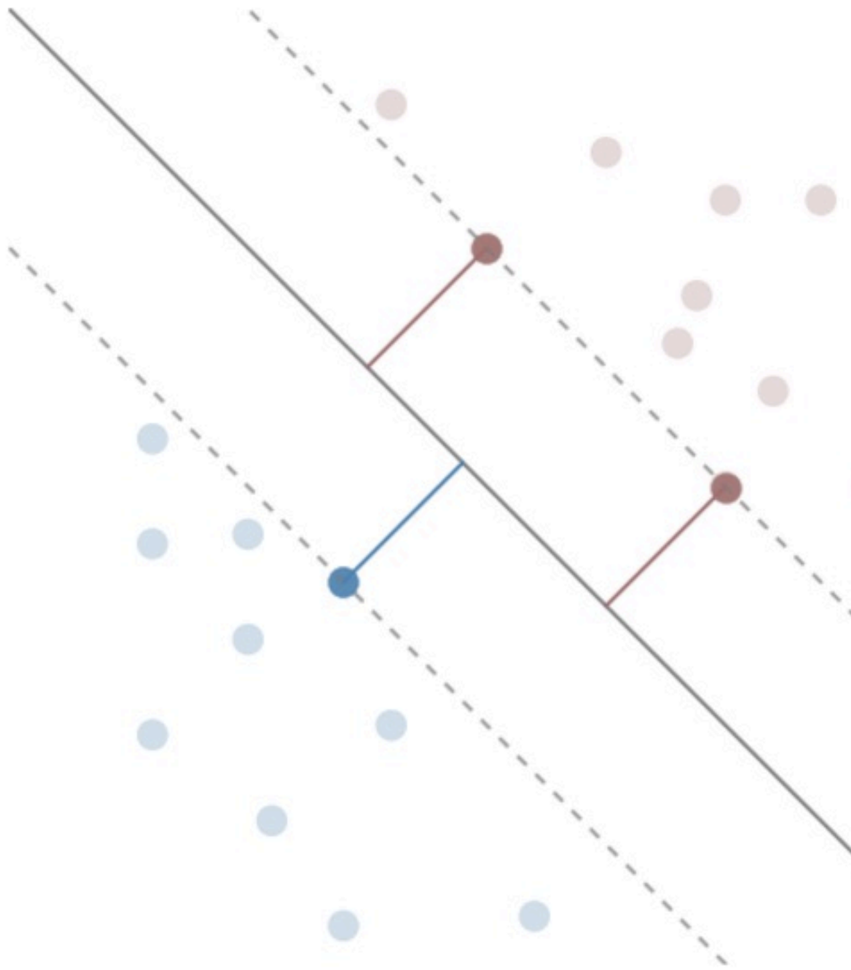
SVM Intuition

Do this by maximizing distance from DB to *closest* training points



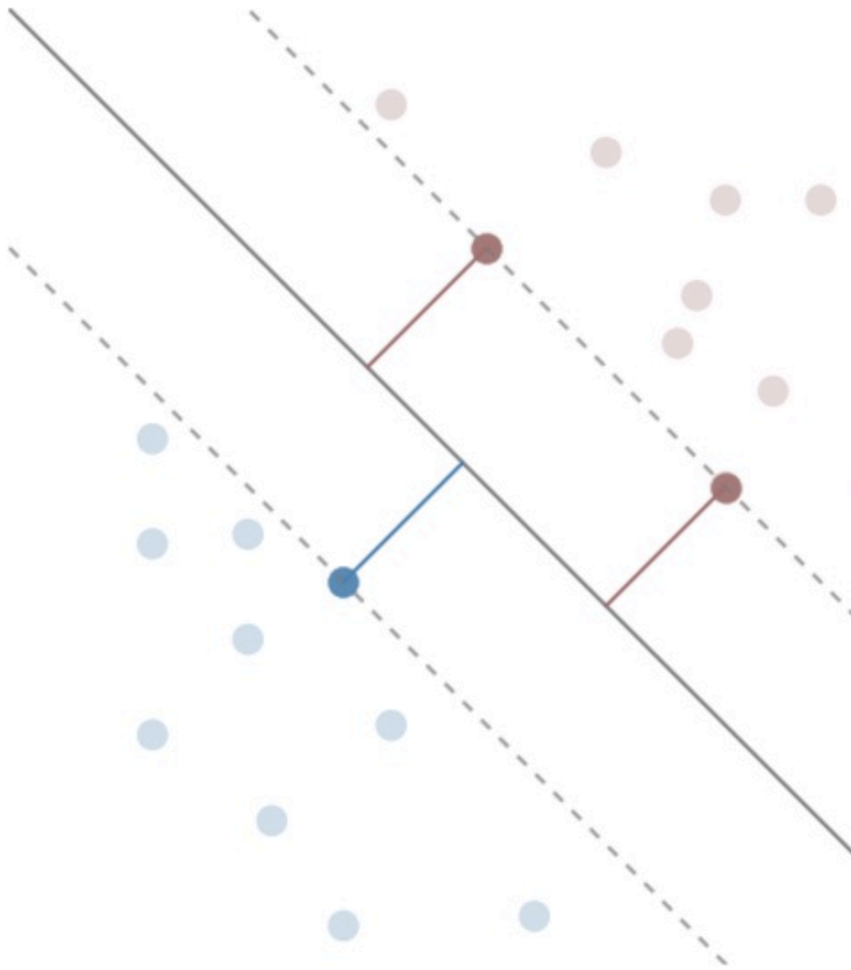
SVM Intuition

This closest distance is called the **Margin**



SVM Intuition

Points closest to DB are called **support vectors**



Maximum Margin Classifier

How do we do this mathematically?

Let the margin be M

Goal:

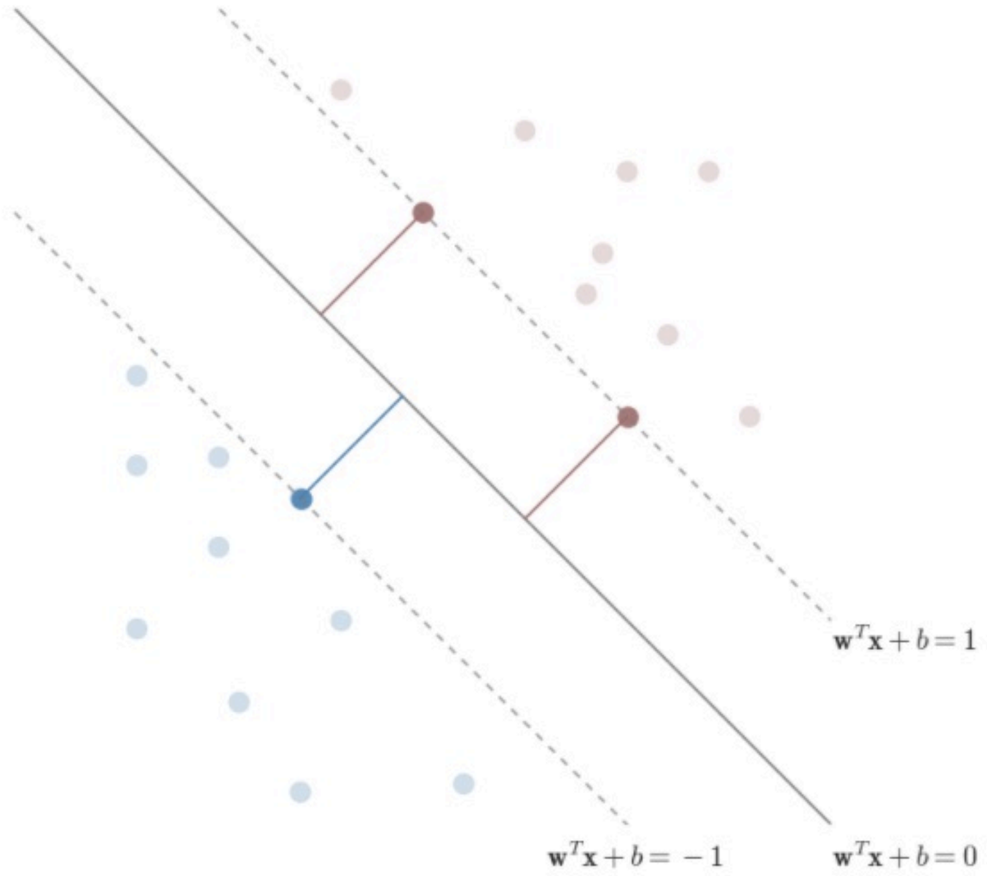
$$\max_{\mathbf{w}, b} \quad M$$

s.t. training points are classified correctly

Maximum Margin Classifier

How do we represent M mathematically?

First, represent support vector boundaries as $\mathbf{w}^T\mathbf{x} + b = \pm 1$



Maximum Margin Classifier

Does this make sense?

We can probably agree that they should look like $\mathbf{w}^T\mathbf{x} + b = \pm K$

Notice that DB $\mathbf{w}^T\mathbf{x} + b = 0$ is unaffected by scalings of $\mathbf{w}$ and b

$$(2\mathbf{w})^T\mathbf{x} + 2b = 0 \quad \Leftrightarrow \quad \mathbf{w}^T\mathbf{x} + b = 0$$

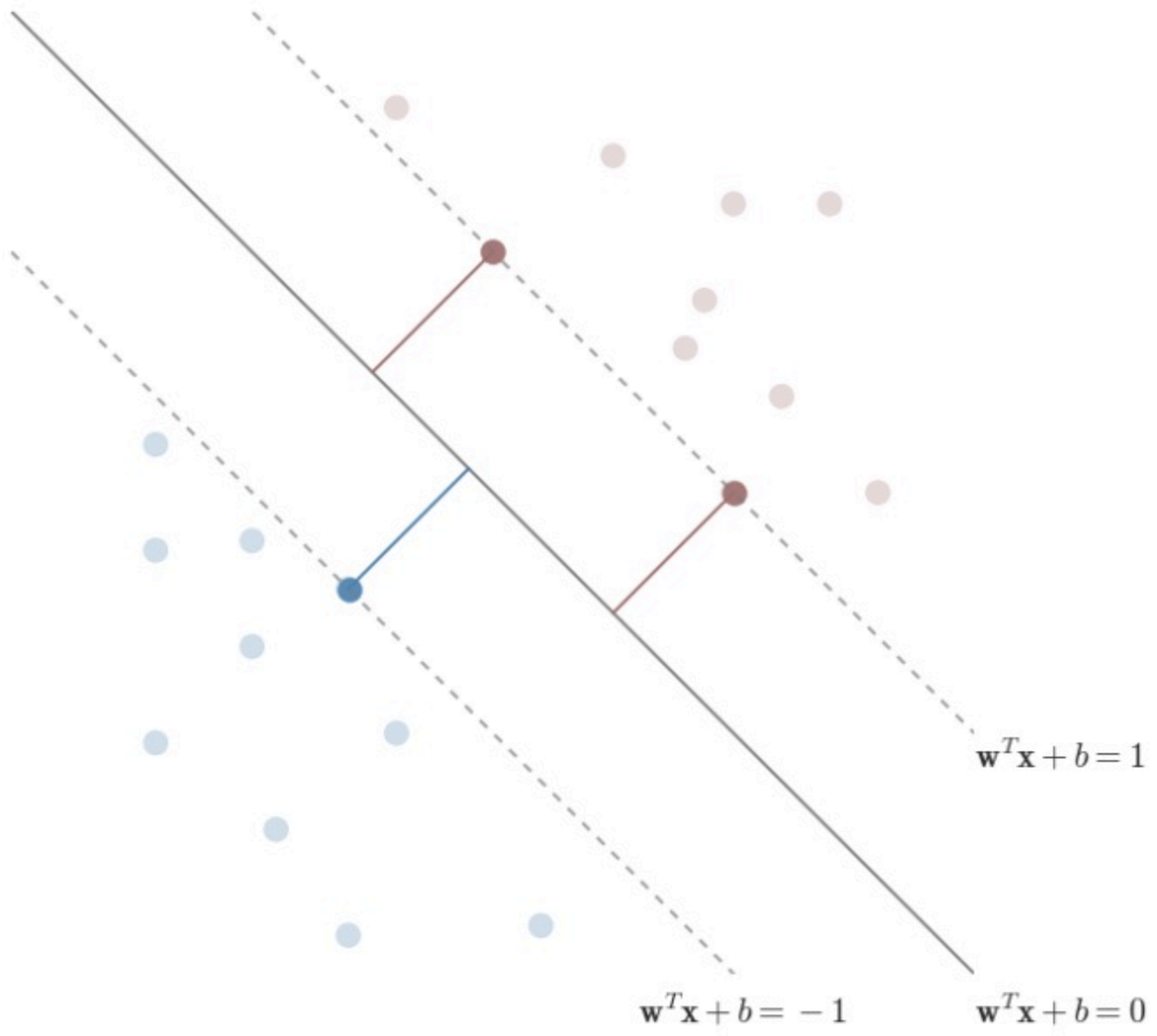
Divide both sides by K

$$\left(\frac{\mathbf{w}}{K}\right)^T\mathbf{x} + \frac{b}{K} = \pm 1$$

Absorb K into $\mathbf{w}$ and $\mathbf{b}$ to get $\mathbf{w}^T\mathbf{x} + b = \pm 1$

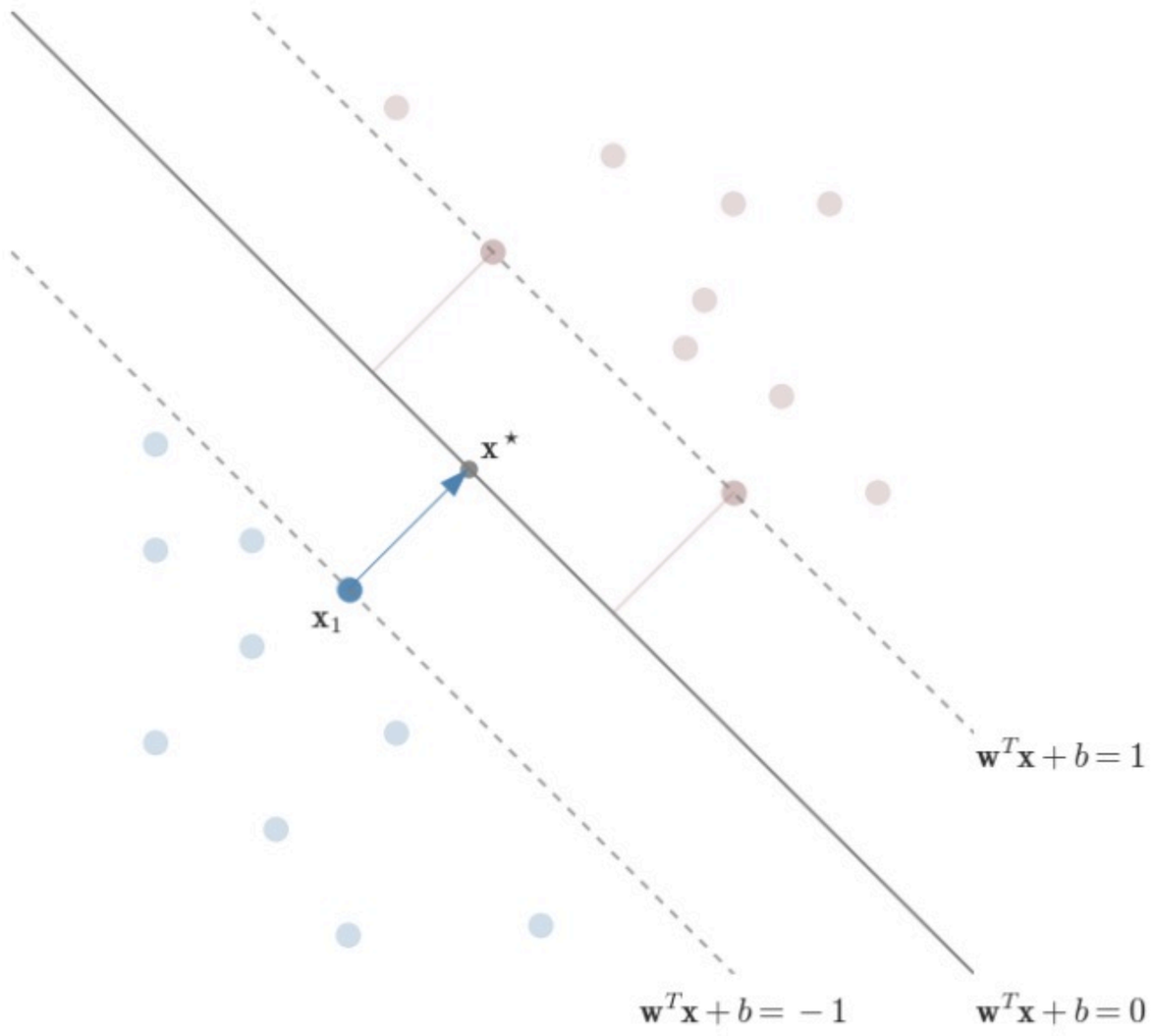
Maximum Margin Classifier

OK, so we agree on the functional forms of the boundaries



Maximum Margin Classifier

To find M we need to do some vector arithmetic



Maximum Margin Classifier

Vector Arithmetic Preparation:

The **norm** of vector $\mathbf{w}$, written as $\|\mathbf{w}\|$, is the magnitude of $\mathbf{w}$

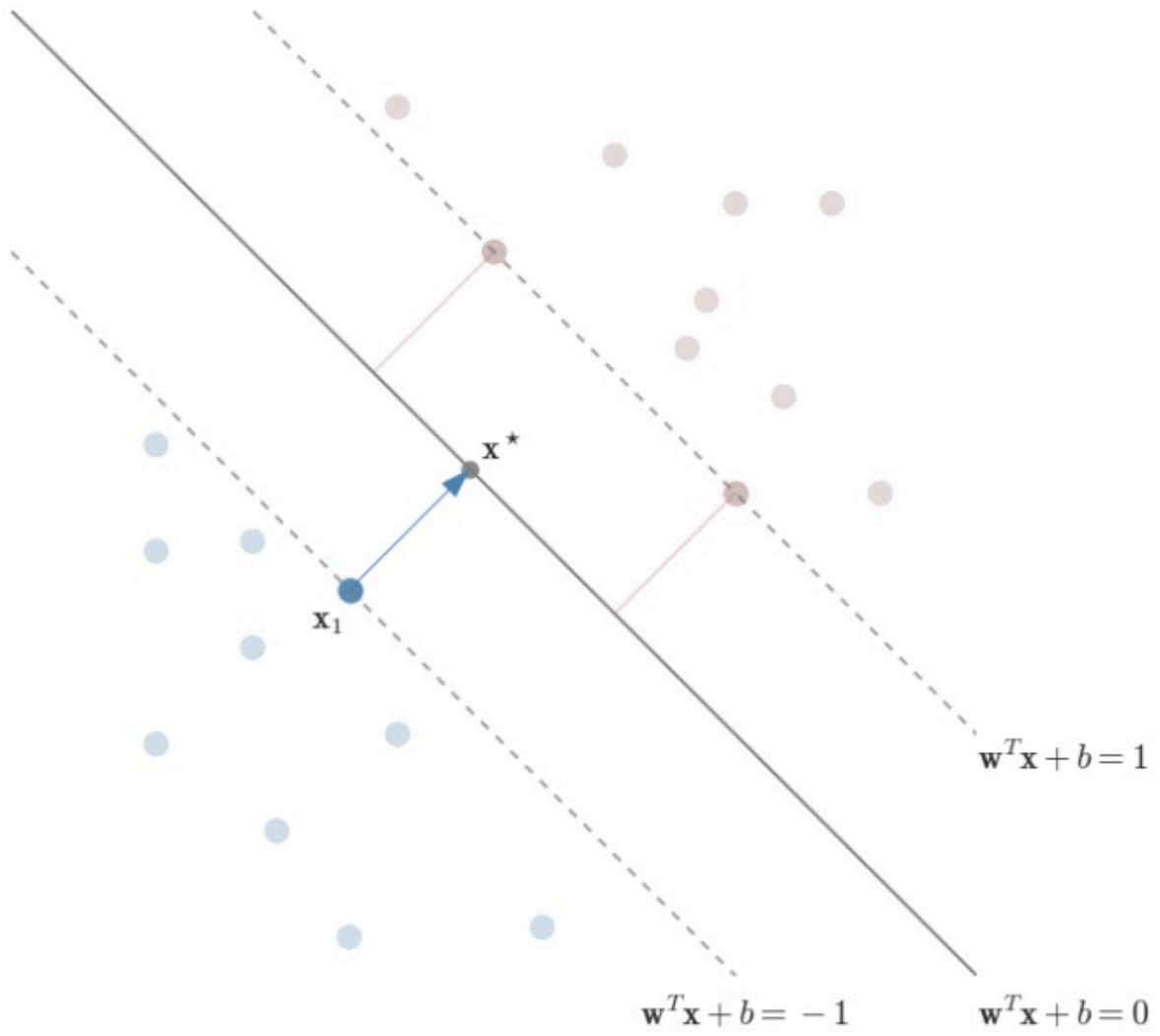
$$\|\mathbf{w}\| = \sqrt{w_1^2 + w_2^2 + \cdots + w_D^2}$$

The dot product of a vector with itself is the square of its norm

$$\|\mathbf{w}\|^2 = w_1^2 + w_2^2 + \cdots + w_D^2 = \mathbf{w}^T\mathbf{w}$$

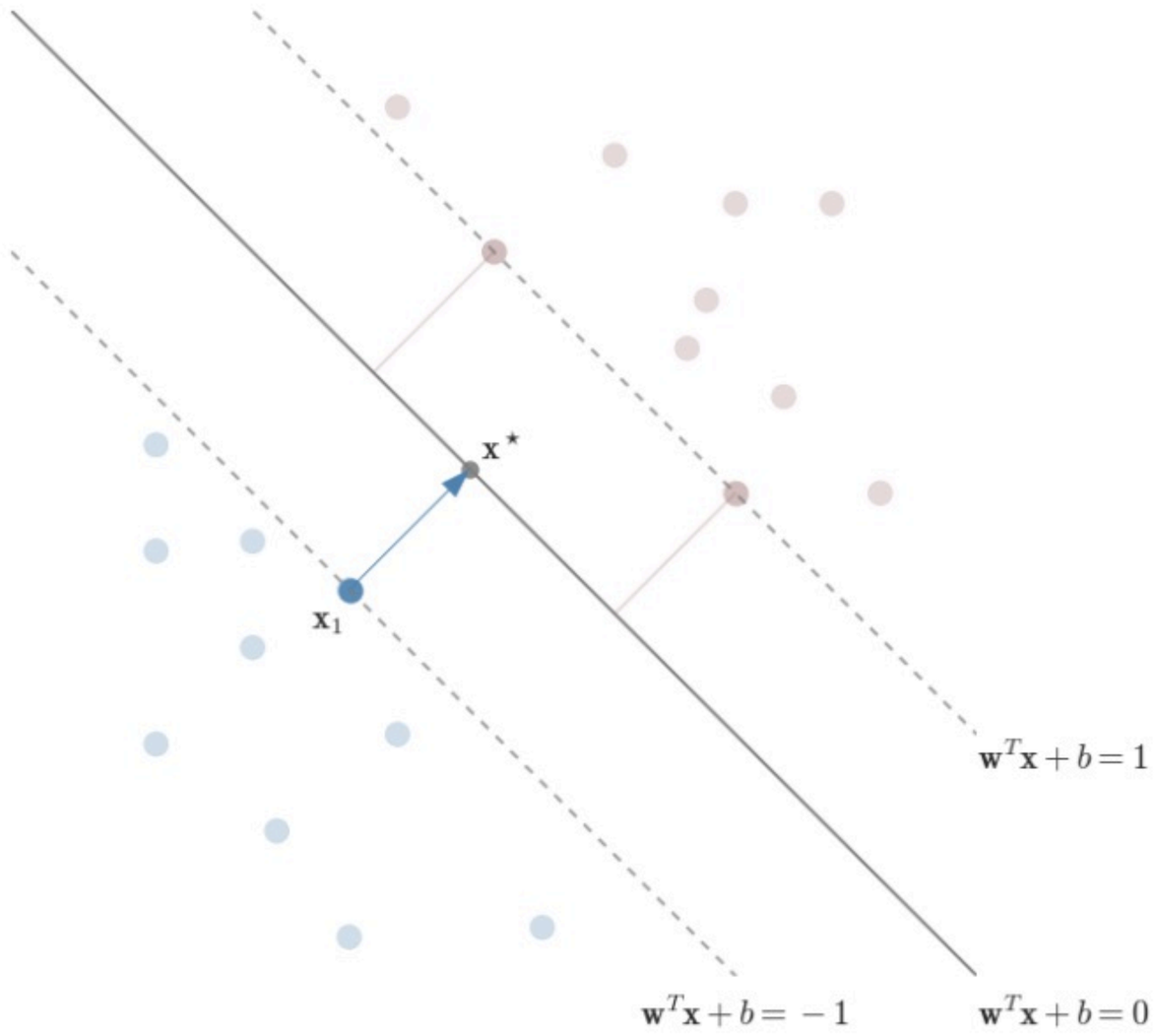
Maximum Margin Classifier

Choose support vector $\mathbf{x}_1$ and its closest point on the DB, $\mathbf{x}^\star$



Maximum Margin Classifier

Get to $\mathbf{x}^\star$ by moving in direction of $\mathbf{w}$: $\mathbf{x}^\star = \mathbf{x}_1 + \lambda \mathbf{w}$



Maximum Margin Classifier

Get to $\mathbf{x}^\star$ by moving in direction of $\mathbf{w}$: $\mathbf{x}^\star = \mathbf{x}_1 + \lambda \mathbf{w}$

The distance moved is the **Margin**

$$M = \|\lambda \mathbf{w}\| = \lambda \|\mathbf{w}\|$$

The point $\mathbf{x}^\star$ is on the DB

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

The support vector $\mathbf{x}_1$ is on the -1 boundary

$$\mathbf{w}^T \mathbf{x}_1 + b = -1$$

Maximum Margin Classifier

$$\mathbf{x}^\star = \mathbf{x}_1 + \lambda \mathbf{w} \quad (1)$$

$$\mathbf{w}^T \mathbf{x}^\star + b = 0 \quad (2)$$

$$\mathbf{w}^T \mathbf{x}_1 + b = -1 \quad (3)$$

Plugging (1) into (2) gives

$$\mathbf{w}^T (\mathbf{x}_1 + \lambda \mathbf{w}) + b = 0 \quad \Rightarrow \quad \mathbf{w}^T \mathbf{x}_1 + b + \lambda \mathbf{w}^T \mathbf{w} = 0 \quad (4)$$

Plugging (3) into (4) gives

$$\lambda \|\mathbf{w}\|^2 = 1 \quad (5)$$

Recalling that $M = \lambda \|\mathbf{w}\|$ and (5) gives $M = \dfrac{1}{\|\mathbf{w}\|}$

Maximum Margin Classifier

Mathier Goal:

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

$$\text{s.t.} \quad \text{training points are classified correctly}$$

The constraints can be written as

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ for points with } y_i = 1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \text{ for points with } y_i = -1$$

Combine to get $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$ for $i = 1, \ldots, m$

Maximum Margin Classifier

Mathiest Goal:

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

$$\text{s.t.} \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

Solve optimization problem to find weights $\mathbf{w}$ and bias b for maximum margin

Note: Rigid enforcement of condition that all training points are classified correctly makes this the **Hard-Margin** SVM.

Problem: This optimization problem is *nasty*

Maximum Margin Classifier

Possibilites:

$$\max_{\mathbf{w},\, b} \frac{1}{\|\mathbf{w}\|} \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \text{(not differentiable)}$$

$$\min_{\mathbf{w},\, b} \|\mathbf{w}\| \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \text{(not differentiable)}$$

$$\min_{\mathbf{w},\, b} \|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \text{(convex + differentiable)}$$

We'll be using the last of the three (with slight modification)

Convex quadratic program = tons of optimized canned software

We'll talk about a specific algorithm next time (SMO)

Convex Opt. with Linear Inequality Constraints

Consider the following general problem:

$$\begin{aligned} \min_{w} \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0 \quad i = 1, \ldots, m \end{aligned}$$

Define the Lagrangian

$$L(w, \alpha) = f(w) + \sum_{i=1}^{m} \alpha_i g_i(w), \quad \text{s.t.} \quad \alpha_i \geq 0$$

Notice that

$$\max_{\alpha} L(w, \alpha) = \begin{cases} f(w) & \text{if } w \text{ is feasible} \\ \infty & \text{if } w \text{ is infeasible} \end{cases}$$

Convex Opt. with Linear Inequality Constraints

Original problem then becomes

$$\min_{w} \max_{\alpha} L(w, \alpha) = \min_{w} \max_{\alpha} f(w) + \sum_{i=1}^{m} \alpha_i g_i(w), \quad \text{s.t.} \quad \alpha_i \geq 0$$

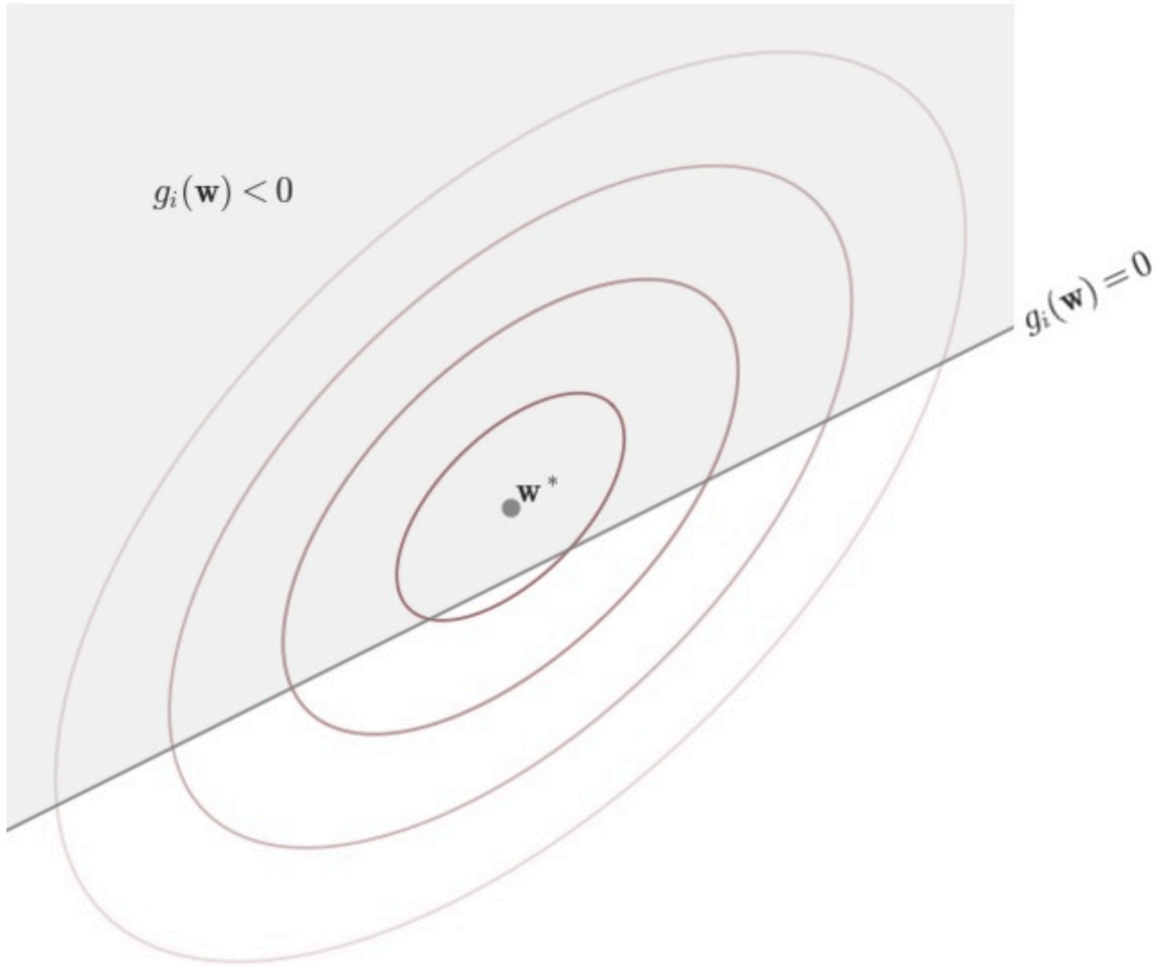
Problem is convex

Exactly one global minimum at

$$\nabla_w L(w, \alpha) = 0 \text{ and } \nabla_\alpha L(w, \alpha) = 0$$

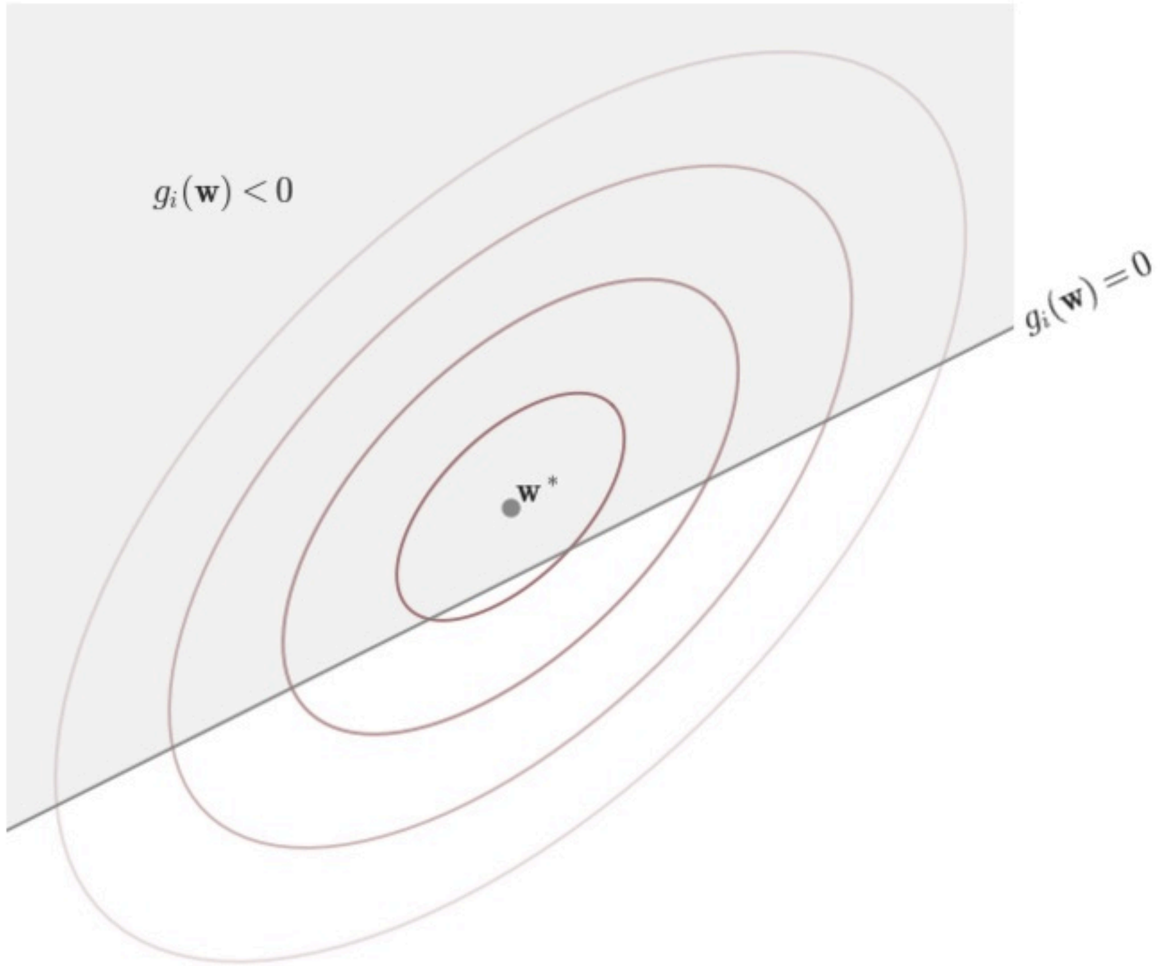
Active and Inactive Constraints

Consider case when w^* that minimizes $f(w)$ is feasible



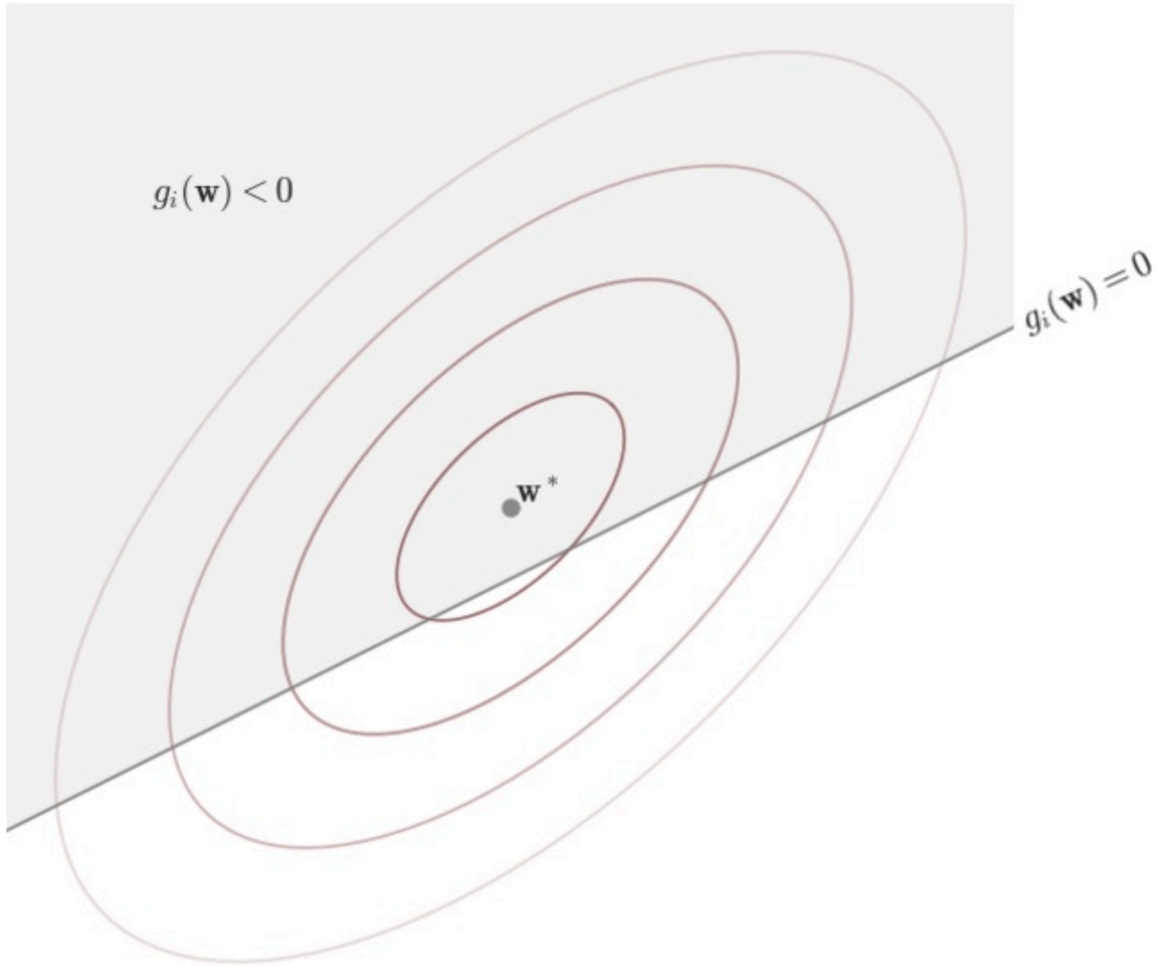
Active and Inactive Constraints

We say that $g_i(w)$ is **inactive** and can set $\alpha_i = 0$



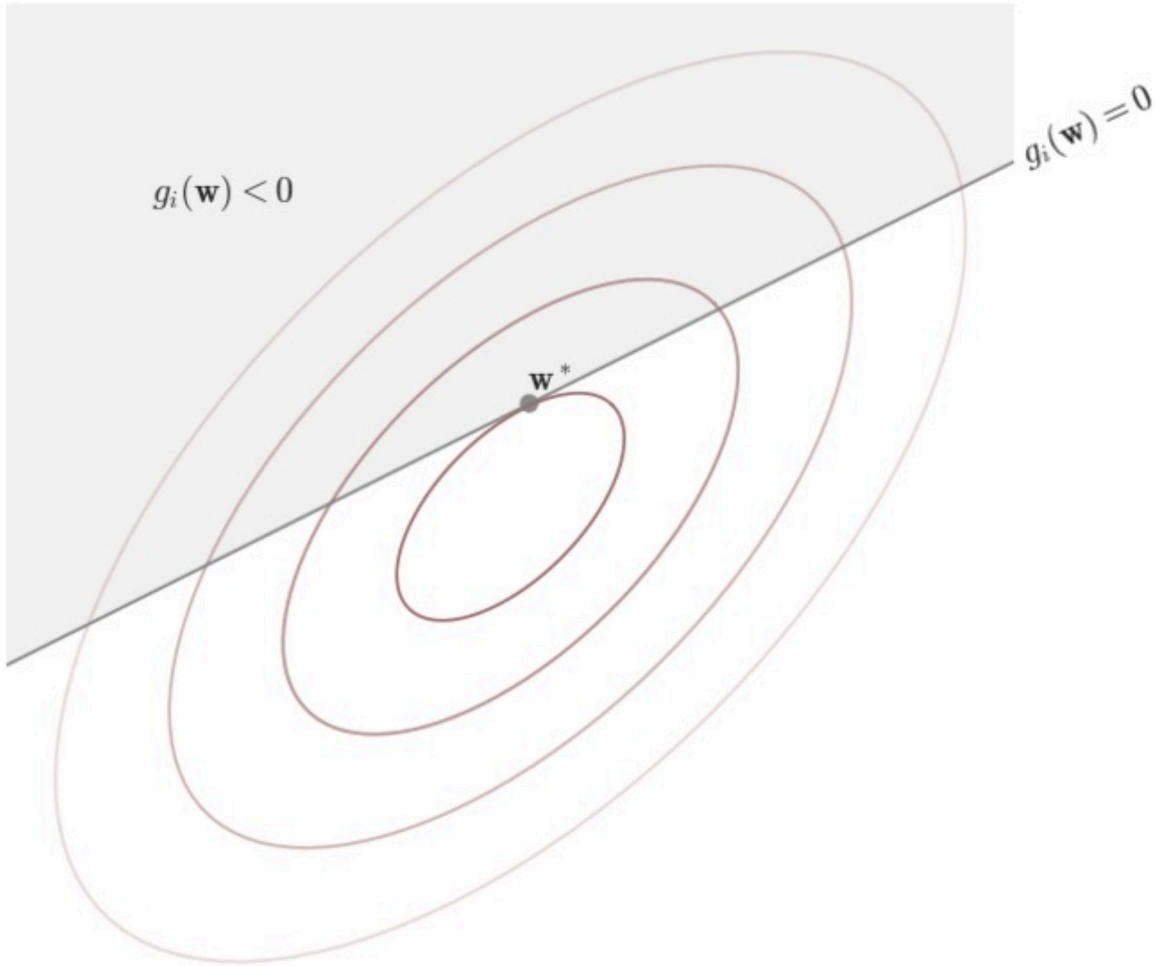
Active and Inactive Constraints

Minimizer is w^* satisfying $\nabla_w f(w) = \mathbf{0}$



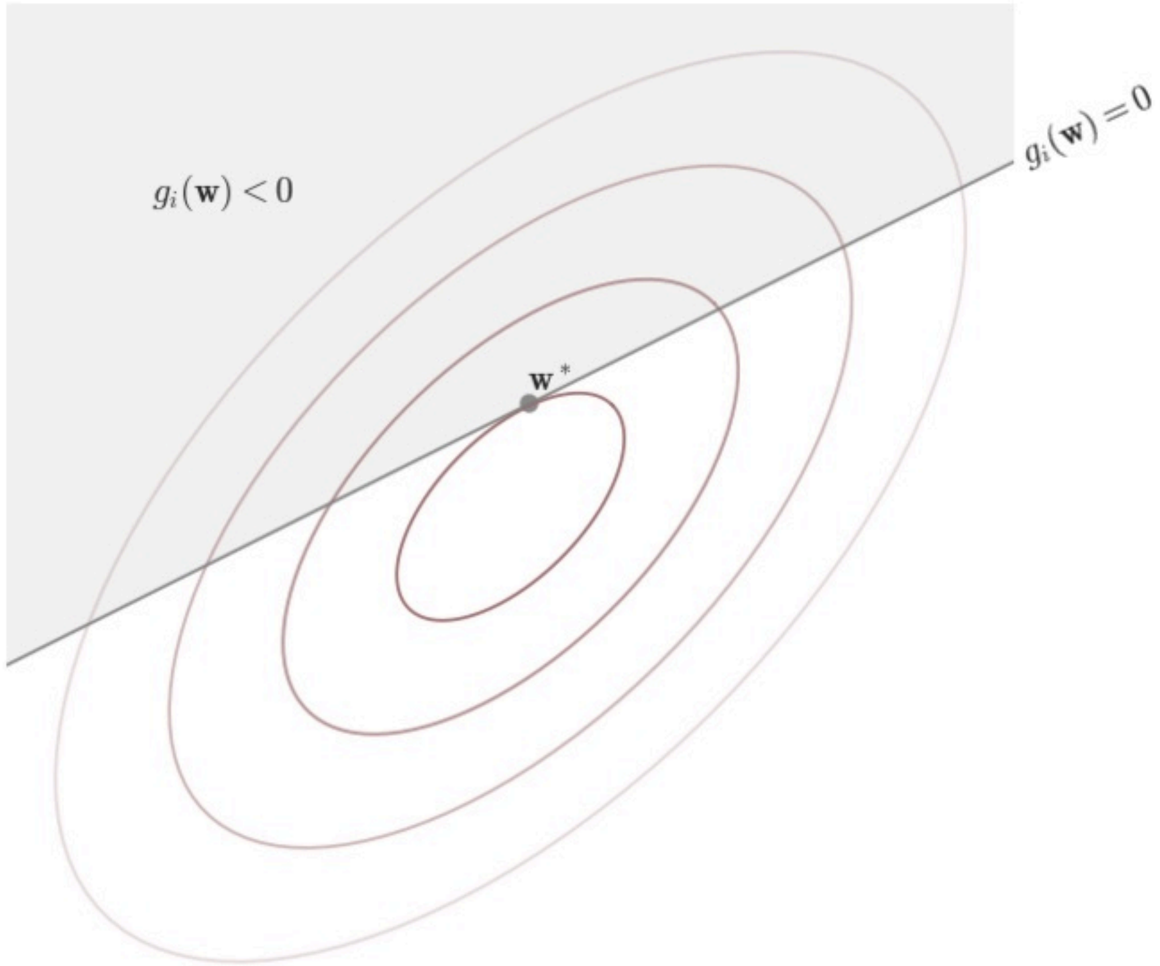
Active and Inactive Constraints

Consider case when w that minimizes $f(w)$ is infeasible



Active and Inactive Constraints

We say that $g_i(w)$ is **active** and will have $\alpha_i > 0$



Active and Inactive Constraints

These two ideas together give us a new condition

Complementary Slackness Condition:

$$\alpha_i g_i(w^*) = 0, \quad i = 1, \dots, m$$

If the minimizer is on an inequality boundary then $\alpha_i > 0$

If the minimizer is not on an inequality boundary then $\alpha_i = 0$

This will give us nice properties in SVM later

Primal vs. Dual Formulation

Current problem is

$$\begin{aligned} \min_{w} \max_{\alpha} \quad & L(w, \alpha) \\ \text{s.t.} \quad & \alpha_i g_i(w) = 0 \\ & \alpha_i \geq 0 \end{aligned}$$

Mathematical Fact: $\max_{\alpha} \min_{w} L(w, \alpha) \leq \min_{w} \max_{\alpha} L(w, \alpha)$

Right-hand side is called the **Primal** problem

Left-hand side is called the **Dual** problem

If Primal is convex then sides are equal at the optimal solution

The Dual Formulation

Dual problem has some nice properties for SVMs, so we'll solve that

$$\max_{\alpha} \min_{\mathbf{w}, b} \quad L(\mathbf{w}, b, \alpha)$$

Putting our problem into the general framework

$$\max_{\alpha} \min_{\mathbf{w}, b} \quad \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i [1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)]$$

The Dual Formulation

Do the inner minimization by setting derivatives to zero

$$\max_{\alpha} \left(\min_{\mathbf{w}, b} \quad \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i [1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)] \right)$$

$$\frac{\partial L}{\partial w_j} = w_j + \sum_{i=1}^{m} -\alpha_i y_i x_{ij} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \alpha_i y_i = 0$$

Plug these into Lagrangian

The Dual Formulation

The Lagrangian at the optimal $\mathbf{w}$ and b becomes

$$\frac{1}{2}\left\|\sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i\right\|^2 + \sum_{i=1}^{m}\alpha_i\left[1 - y_i\left(\left(\sum_{j=1}^{m}\alpha_j y_j \mathbf{x}_j^T\right)\mathbf{x}_i + b\right)\right]$$

$$= \frac{1}{2}\left\|\sum_{i}\alpha_i y_i \mathbf{x}_i\right\|^2 - \sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_j^T \mathbf{x}_i + \sum_{i}\alpha_i - b\sum_{i}\alpha_i y_i$$

$$= \sum_{i}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_j^T \mathbf{x}_i$$

The Dual Formulation

The dual problem then becomes

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j^T \mathbf{x}_i \\ \text{s.t.} \quad & \alpha_i > 0 \text{ if } y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1 \\ & \alpha_i = 0 \text{ if } y_i(\mathbf{w}^T \mathbf{x}_i + b) > 1 \end{aligned}$$

Lots of cool things to notice...

- Objective function only depends on $\mathbf{x}_j^T \mathbf{x}_i$
- Super convenient when we go to nonlinear SVM

The Dual Formulation

Recall that $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$

- Weight vector only depends on $\alpha_i \neq 0$
- These are exactly the support vectors

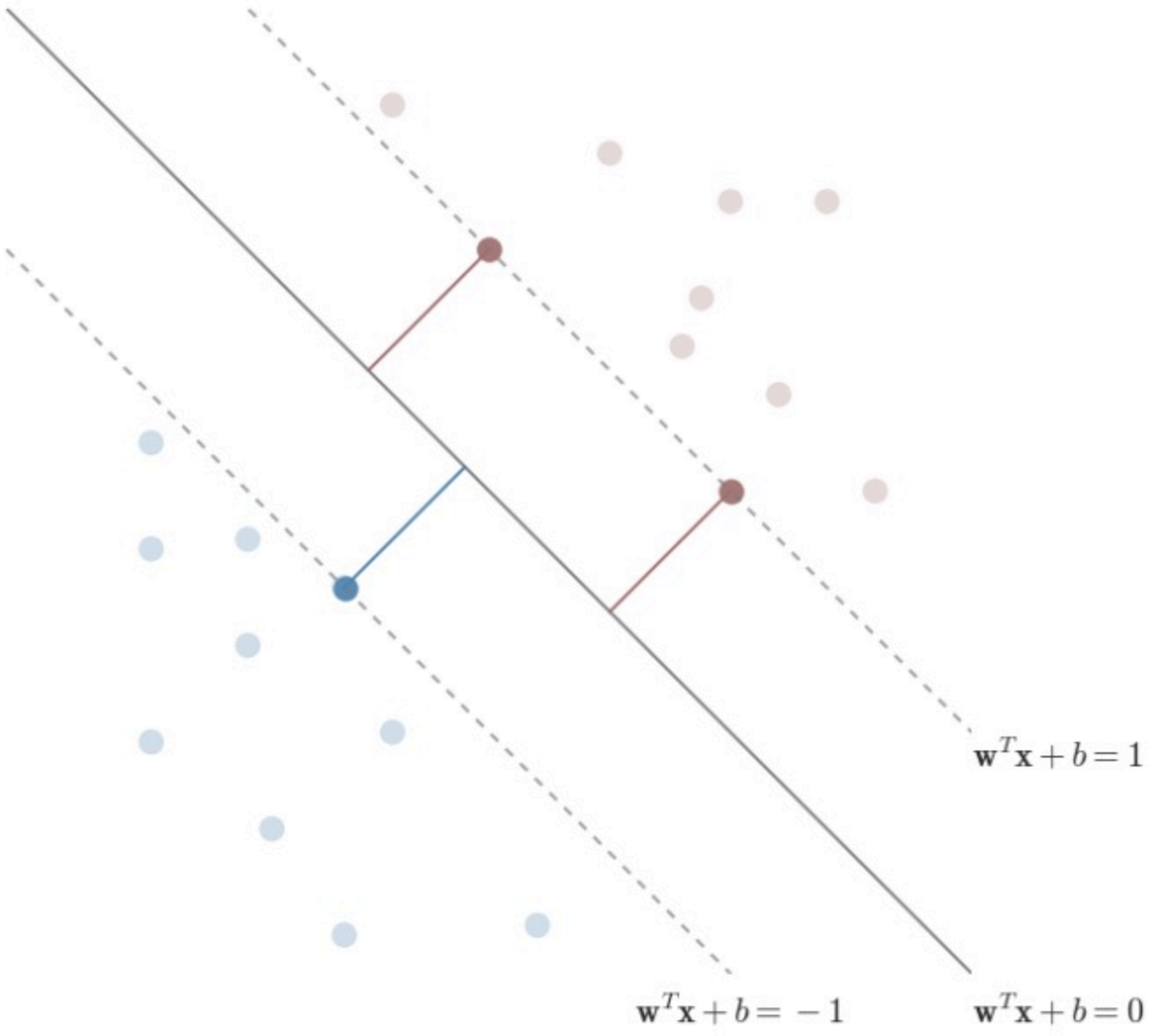
How do we classify a new point $\mathbf{x}$?

$$\mathbf{w}^T \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Again, only involves support vectors

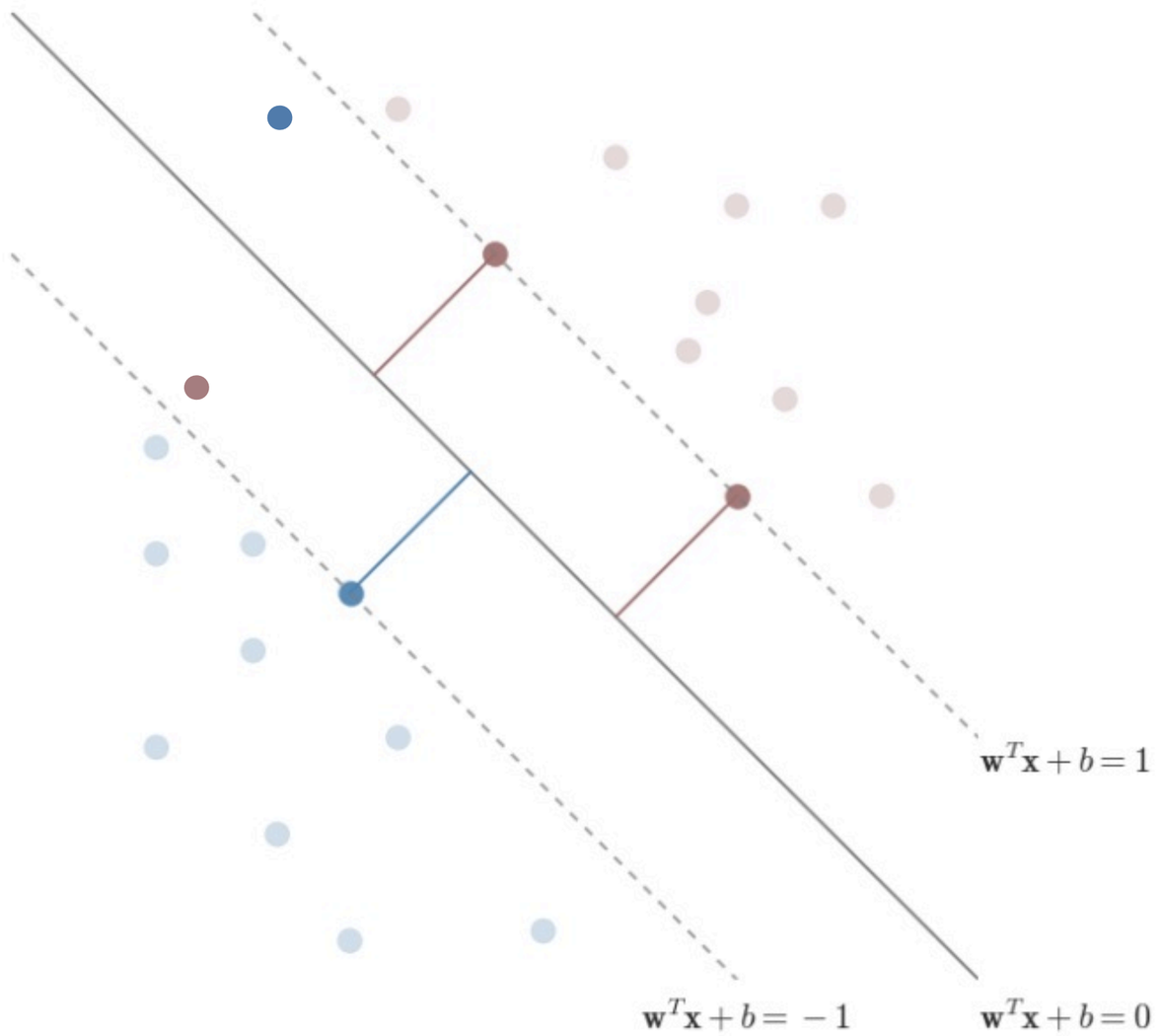
Fixing Problematic Assumptions

Recall that we assumed data was linearly separable



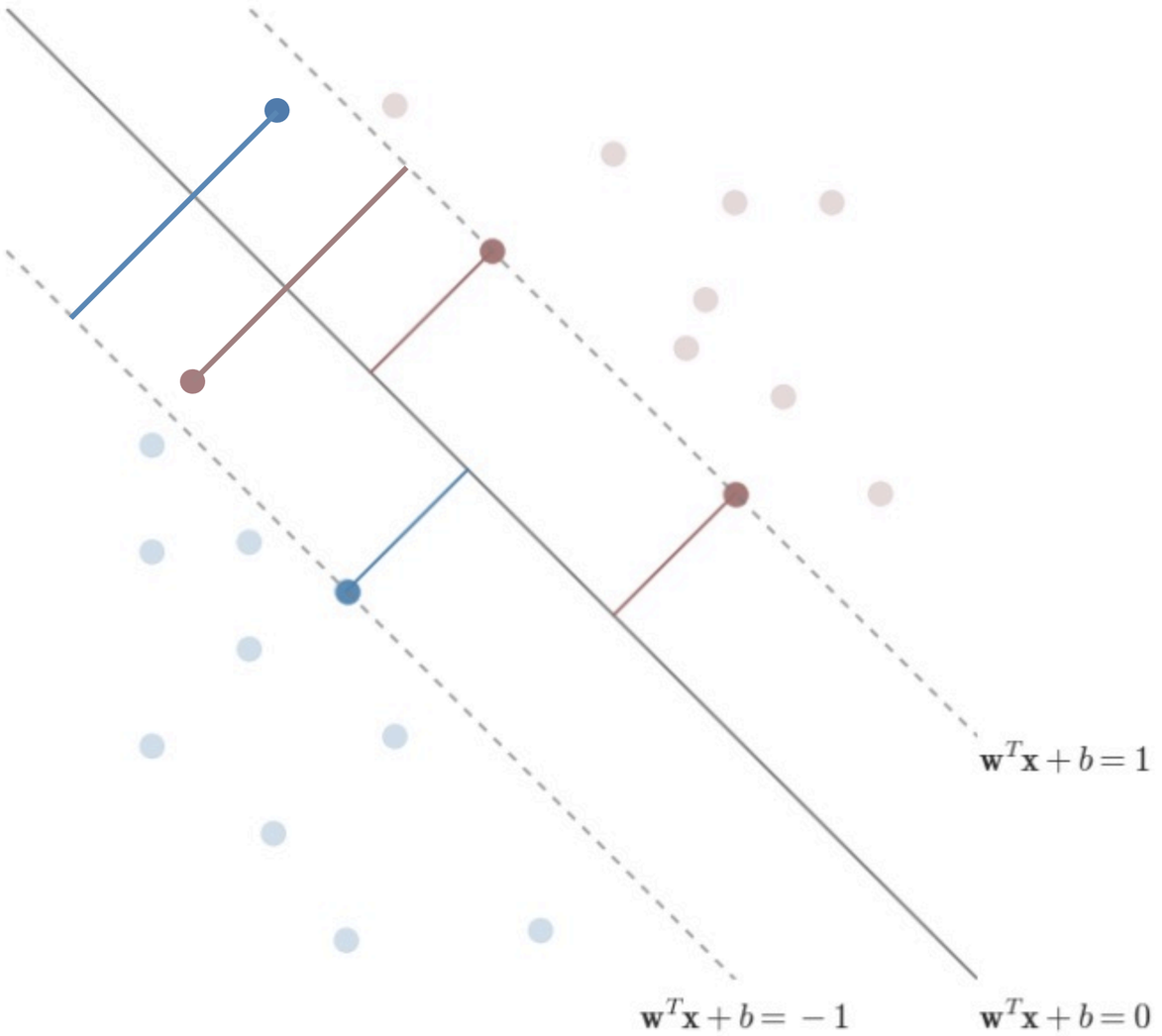
Fixing Problematic Assumptions

But what if it's not?



Fixing Problematic Assumptions

Introduce slack variables, allow for some misclassification



Coming Up

- Slack variables and the Soft-Margin SVM
- The SMO algorithm
- The Kernel Trick for nonlinear classification
- Theoretical properties of SVMs

In Class

In Class

In Class

In Class

In Class

In Class

In Class