Support Vector Machines

CS 480
Intro to Artificial Intelligence

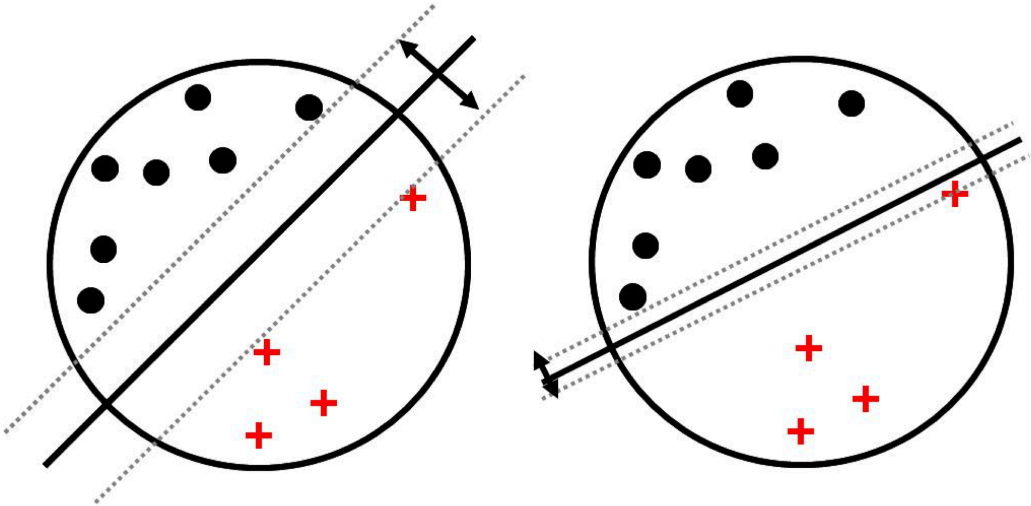
The “large margin” principle

Another basis for picking the “best” hypothesis

Assume the data is linearly separable, which **line** is the best choice for the decision boundary?

If we believe our training data has some inherent noise, we should pick the line that’s as far away from the data as possible. That way any small errors in the **x**’s are less likely to move from one side of the line to the other.

This is the **large margin** principle: The “best” hypothesis is the one that **maximizes** the distance to the training data



Geometry of the decision boundary

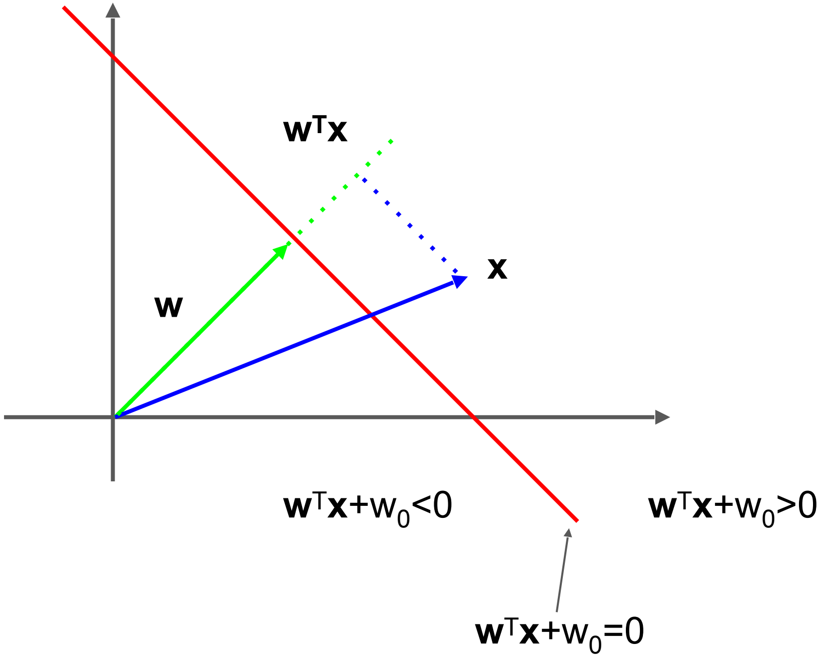
For now: binary classification with labels {-1,1}

We can describe a linear decision boundary with a vector, *w*, that's perpendicular to it:

$$\mathcal{H} = \{h_{\mathbf{w},w_0} : h_{\mathbf{w},w_0}(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^\top \mathbf{x} + w_0)\}$$

$$\mathrm{sgn}(z) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{otherwise} \end{cases}$$

This hypothesis **projects** the point **x** onto the vector **w** and checks to see what side of the boundary it falls on



$\mathbf{w}^T\mathbf{x}$
$\mathbf{x}$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x}+w_0<0$
$\mathbf{w}^T\mathbf{x}+w_0>0$
$\mathbf{w}^T\mathbf{x}+w_0=0$

Geometry of the margin

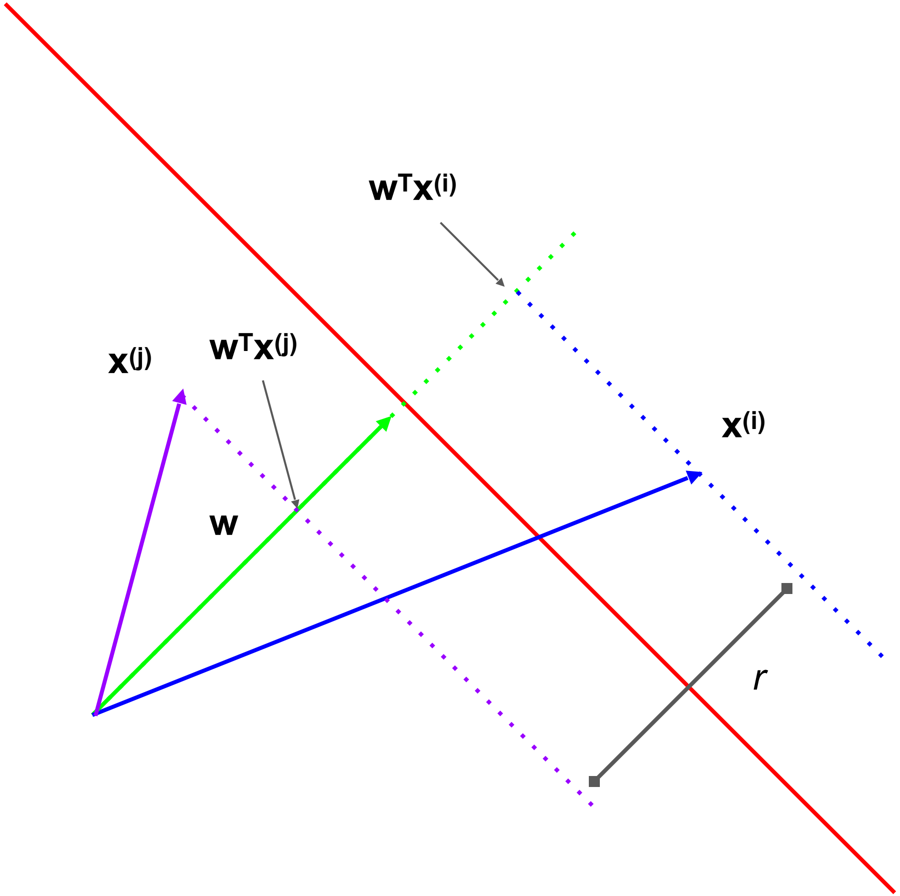
Let **x**(i) be the point closest to the boundary in the positive region, and **x**(j) be closest in the negative region, then the margin, *r*, is

$$r = (\mathbf{w}^\top \mathbf{x}^{(i)} + w_0) - (\mathbf{w}^\top \mathbf{x}^{(j)} + w_0)$$

$$r = \mathbf{w}^\top (\mathbf{x}^{(i)} - \mathbf{x}^{(j)})$$

$$\frac{r}{\|\mathbf{w}\|} = \hat{\mathbf{w}}^\top (\mathbf{x}^{(i)} - \mathbf{x}^{(j)})$$

Note: If we re-scaled the data (multiplied each **x** by some constant), this wouldn't change the **direction** of **w**.



$\mathbf{w}^T\mathbf{x}^{(i)}$
$\mathbf{x}^{(j)}$
$\mathbf{w}^T\mathbf{x}^{(j)}$
$\mathbf{x}^{(i)}$
$\mathbf{w}$
r

Finding the maximum margin - primal

We want to find **w** that maximizes the margin

$$\arg\max_{\mathbf{w}} \frac{r}{\|\mathbf{w}\|} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ s.t. constraints}$$

Where the constraints make sure all positive and negative examples end up on the correct side of the line

$$\text{Constraints:}$$

$$y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + w_0) \geq 1$$

$$\forall (\mathbf{x}^{(i)}, y^{(i)}) \in S$$

Finding the maximum margin - dual

This is a **quadratic optimization** problem. Solvers exist, and it can be shown that solving the following problem is equivalent

$$\arg\max_{\alpha} \sum_{i=1}^{N} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} \mathbf{x}^{(i)\top} \mathbf{x}^{(j)}$$

$$\text{Subject to}$$

$$\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} = 0, \quad \alpha^{(i)} \geq 0, \quad \forall \alpha^{(i)}$$

And **w** is computed as

$$\mathbf{w} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}$$

To find w_0, first compute **w**, then plug in a labeled training point to the constraint $y^{(i)}(\mathbf{w}^{\top}\mathbf{x}^{(i)}+w_0)>=1$

Support Vectors

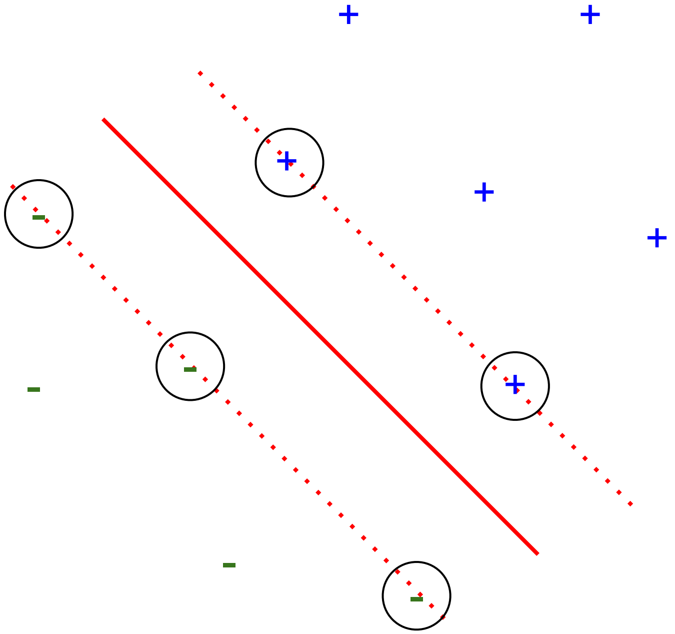
What does $\alpha^{(i)}$ actually mean?

Note: only **some** of the data points have an impact on the decision boundary.

Points that are far away **can not change** which **w** gives the best margin

Points that lie right at the **edge** of the margin **do** determine **w**

We call these points **support vectors**, and they are the **only** points with $\alpha^{(i)}>0$



Soft Margin Classifier

For most data, we cannot enforce the constraint

$$y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + w_0) \geq 1,\ \forall(\mathbf{x}^{(i)}, y^{(i)}) \in S$$

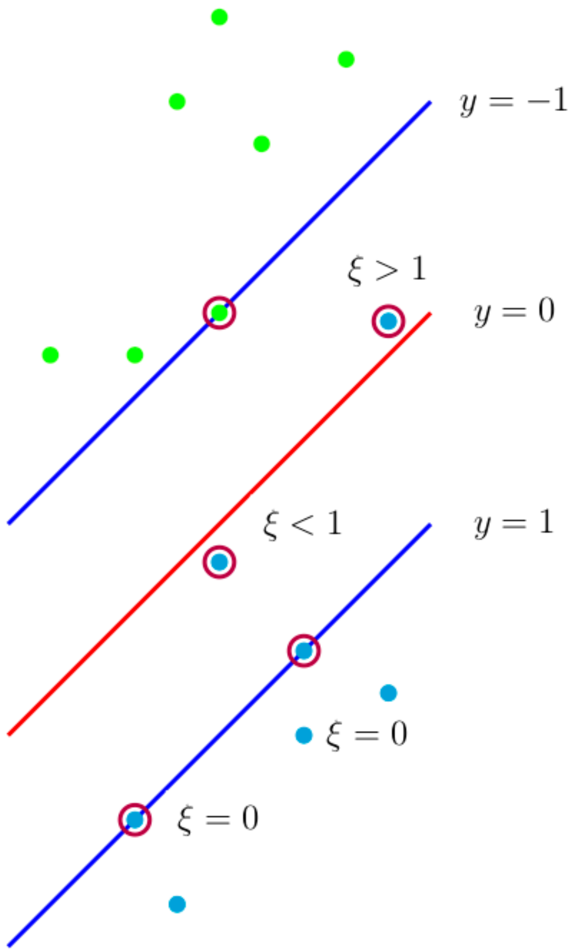
We can modify the optimization to include **slack variables**

$$\arg\min_{\mathbf{w},\xi_i} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

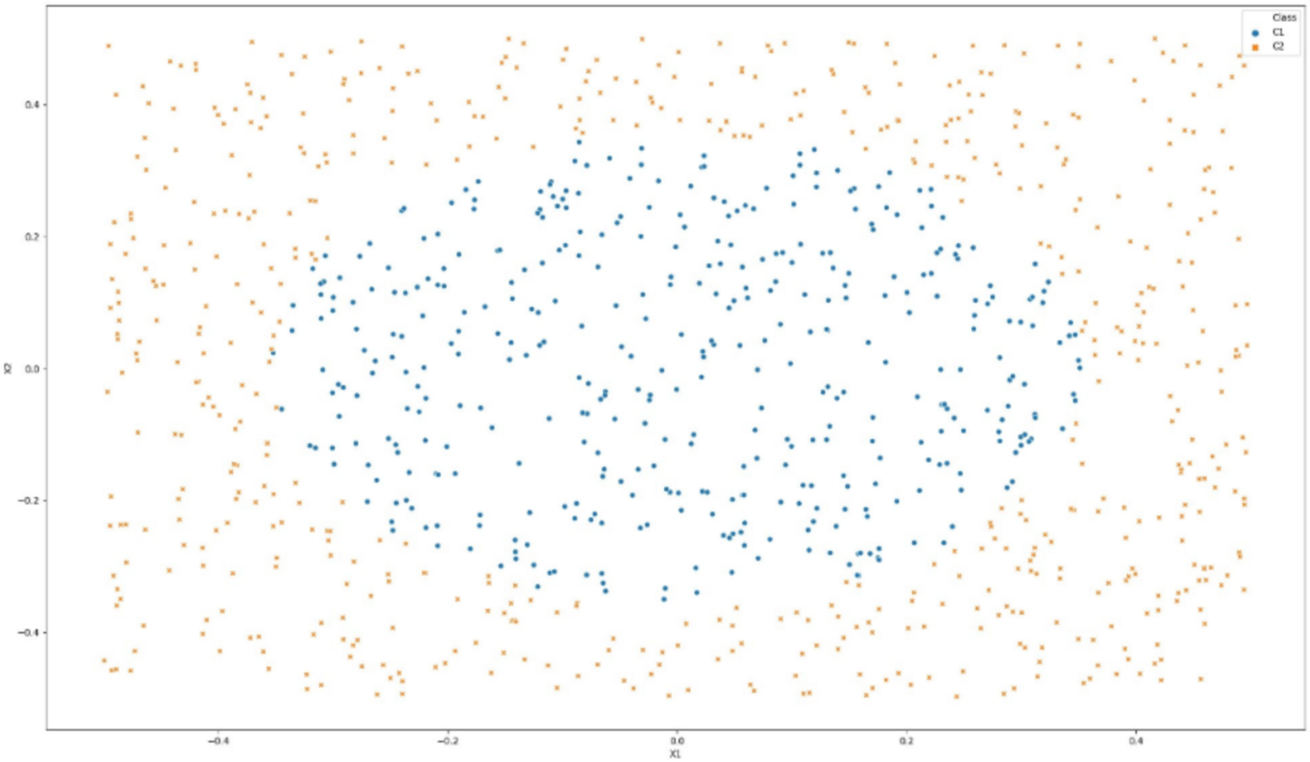
$$\text{subject to}$$

$$y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + w_0) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

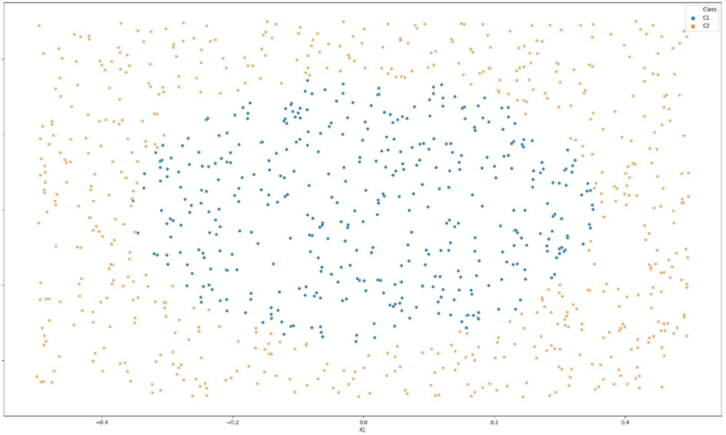
But this only works well if the data is “mostly linearly separable” with a small number of **outliers**

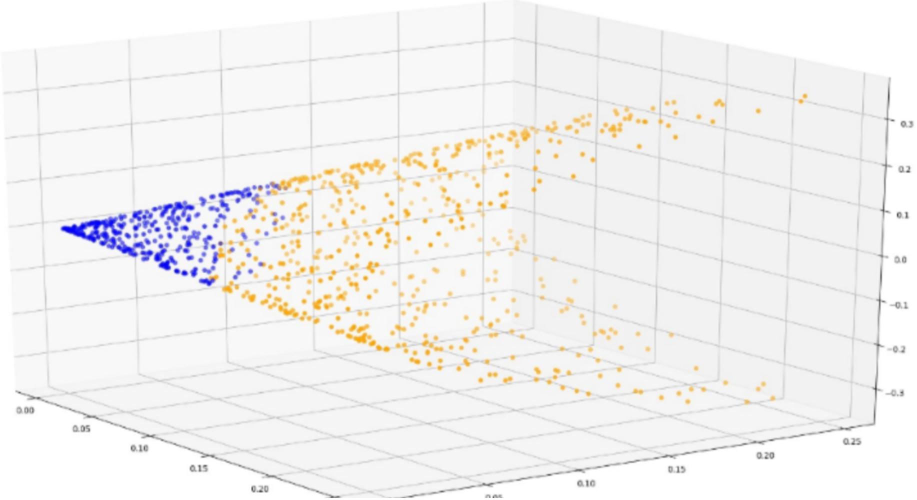


Data that’s **really** not linearly separable



Transforming data to **make** it be linearly separable





$\mathbf{x} = [x_1,x_2]$ → $\phi()$ → $\phi(\mathbf{x}) = [(x_1)^2,(x_2)^2,\sqrt{2}(x_1*x_2)\ ]$

Using basis functions for SVM

Basis function expansion let us use linear regression on nonlinear data

We can use a similar trick to use linear SVMs on non-linearly separable data

In general, by transforming data into **higher dimensions**, it is more likely that some linear boundary exists

$$\alpha = \arg\max_{\alpha} \sum_{i=1}^{N} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} \phi(\mathbf{x}^{(i)})^{\top} \phi(\mathbf{x}^{(j)})$$

Notice: the only place where ϕ appears is in this inner product

Inner products and kernels

Instead of computing ϕ(**x**) for all the data points, what if we had a function that could compute the **inner product** of ϕ(**x**)$^\top\phi$(**x'**) directly?

$$\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = k(\mathbf{x}, \mathbf{x}')$$

We'll call this kind of function a **kernel**, and we can swap it in wherever we see an inner product of the transformed input, ϕ(**x**):

$$\arg\max_{\alpha} \sum_{i=1}^{N} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} \phi(\mathbf{x}^{(i)})^\top \phi(\mathbf{x}^{(j)})$$

$$\arg\max_{\alpha} \sum_{i=1}^{N} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

Kernels everywhere

We can find “kernelized” versions of lots of different methods.

“Kernelized” basis function expansion

$$h_k(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top (K + \lambda I)^{-1} Y$$

$$K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

$$k_n(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}^{(i)})$$

Nadaraya-Watson kernel regression

$$h_k(\mathbf{x}) = \frac{\sum_{i=1}^N k(\mathbf{x}, \mathbf{x}^{(i)}) y^{(i)}}{\sum_{j=1}^N k(\mathbf{x}, \mathbf{x}^{(j)})}$$

And there are **many** different kernels functions we could use

Polynomial kernel

$$k_{\text{poly}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^p$$

Gaussian/RBF kernel

$$k_{\text{RBF}(\sigma)}(\mathbf{x}, \mathbf{x}') = \exp\{-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\}$$

Sigmoidal kernel

$$k_{(a,b)}(\mathbf{x}, \mathbf{x}') = \tanh(a\mathbf{x}^\top \mathbf{x}' + b)$$

What makes a valid kernel?

Not all functions are kernels! Loosely, a valid kernel function has to “behave like an inner product.”

One of the simplest technical definitions: The **Gram matrix**, *K*, has to be positive semidefinite for **any** set of **x**$^{(i)}$

$$K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

$$\mathbf{v}^\top K \mathbf{v} \geq 0, \quad \forall \mathbf{v} \in \mathbb{R}^N$$

Once we have some kernels, we can construct new ones with the following rules:

$$k(\mathbf{x}, \mathbf{x}') = c \cdot k_1(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\{k_1(\mathbf{x}, \mathbf{x}')\}$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$$

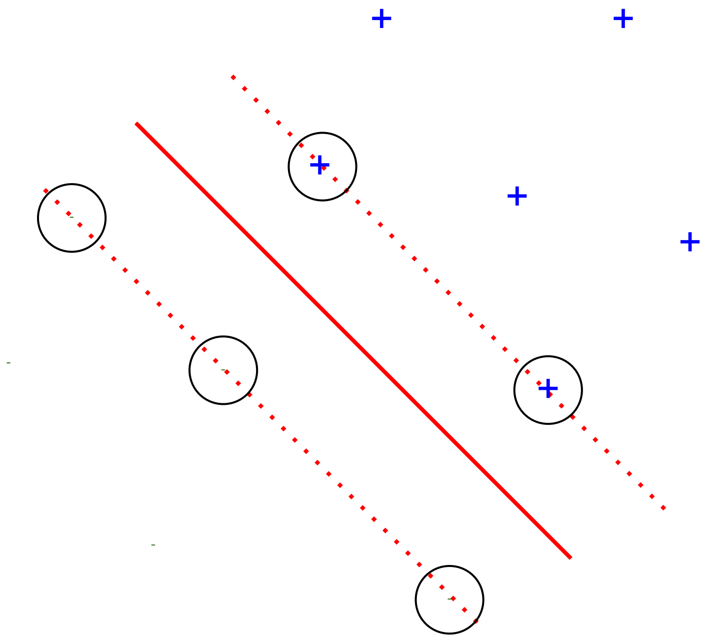
$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top A \mathbf{x}'$$

$$\vdots$$

Why are support vectors important?

We can show that the "kernelized" SVM classifier has the form

$$h(\mathbf{x}) = \text{sgn}\left(w_0 + \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} k(\mathbf{x}^{(i)}, \mathbf{x})\right)$$



Notice
For any of the training data points that are **not** support vectors, we don't have to compute (potentially costly) kernel function!

SVMs are sometimes called **sparse kernel** methods for this reason

Summary and preview

Wrapping up

- Support Vector Machines are a type of classifier that’s based on the **maximum margin** principle
- The parameters of the model are found using quadratic programming methods, which additionally determine which training samples are **necessary** for finding the boundary, the so called “support vectors”
- We can handle non-linearly separable data by using **kernels**