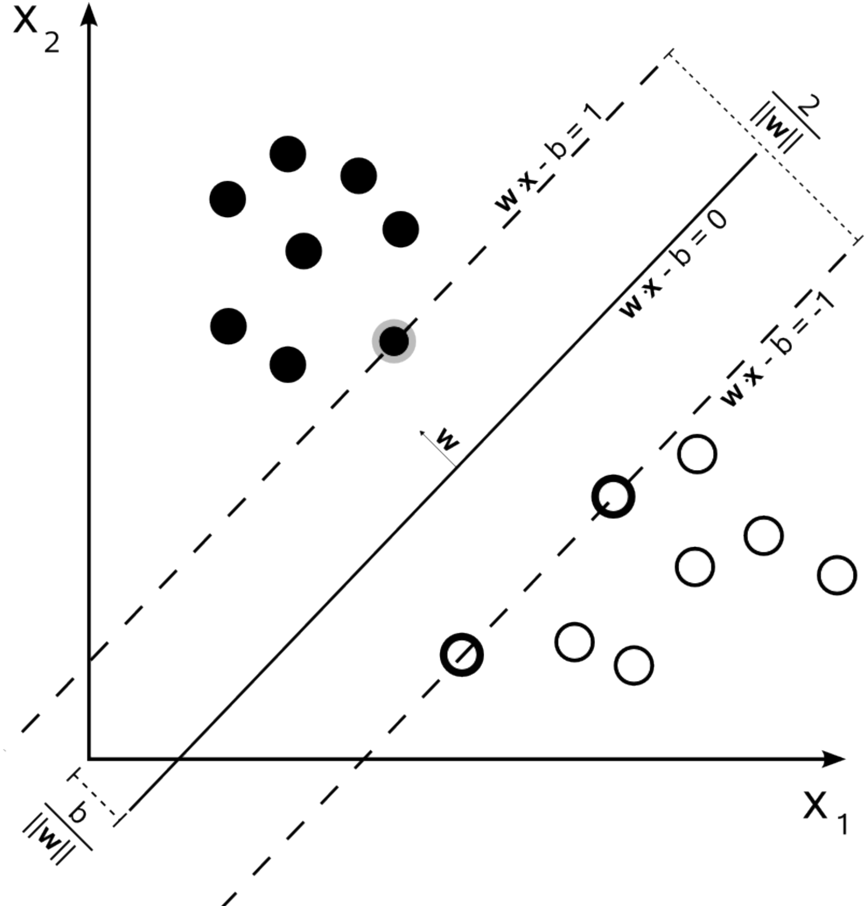
GENERAL ASSEMBLY

DATA SCIENCE
SUPPORT VECTOR MACHINES

Supervised Machine Learning Model

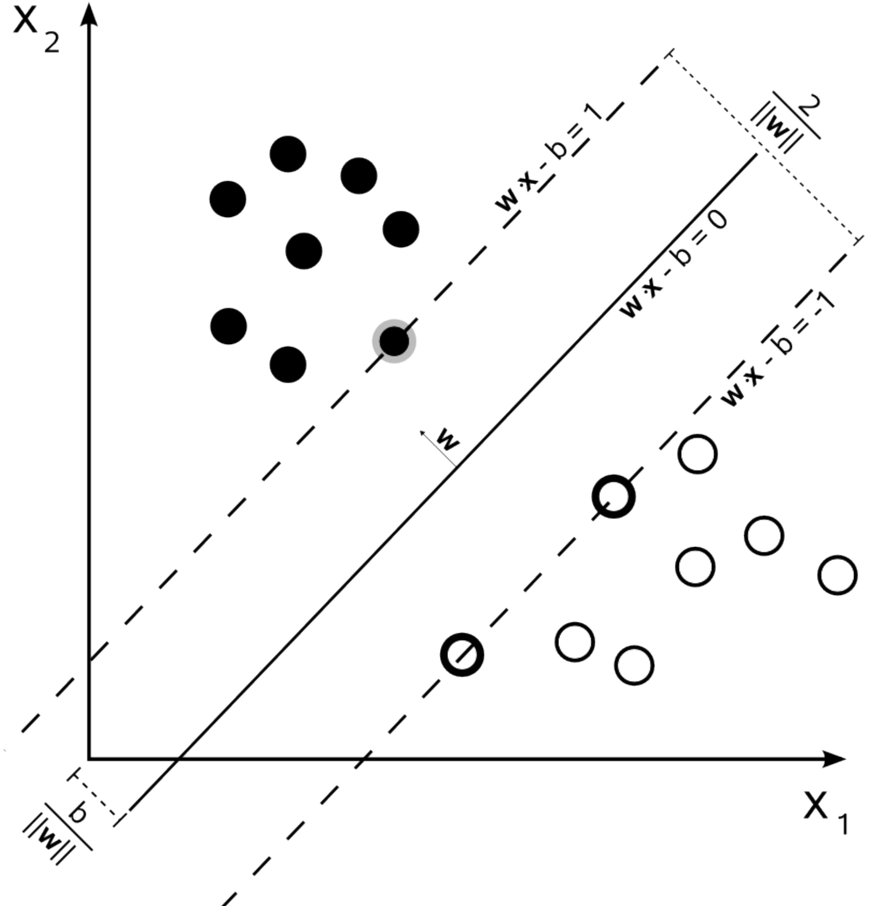
Can be used for regression..

Constructs a hyperplane to separate classes in space

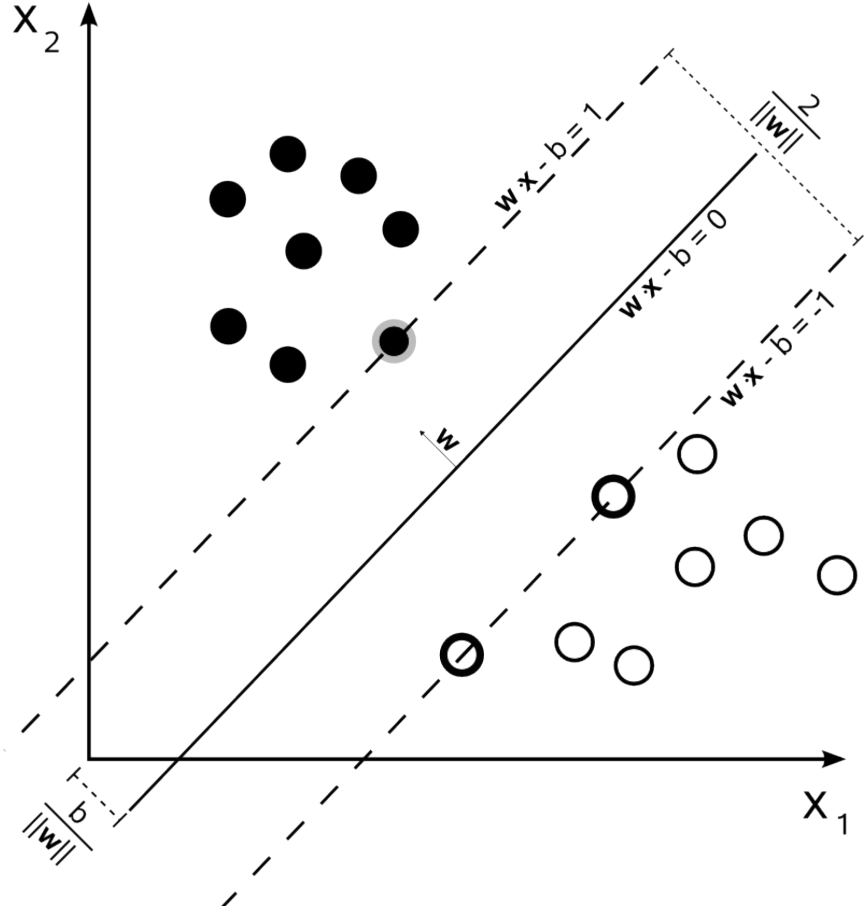


x_2
x_1
$\mathbf{w}\cdot\mathbf{x} - b = 1$
$\mathbf{w}\cdot\mathbf{x} - b = 0$
$\mathbf{w}\cdot\mathbf{x} - b = -1$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}$

The **margin** is the distances between the nearest data points and the hyper-plane.



We want to maximize the width of the margin

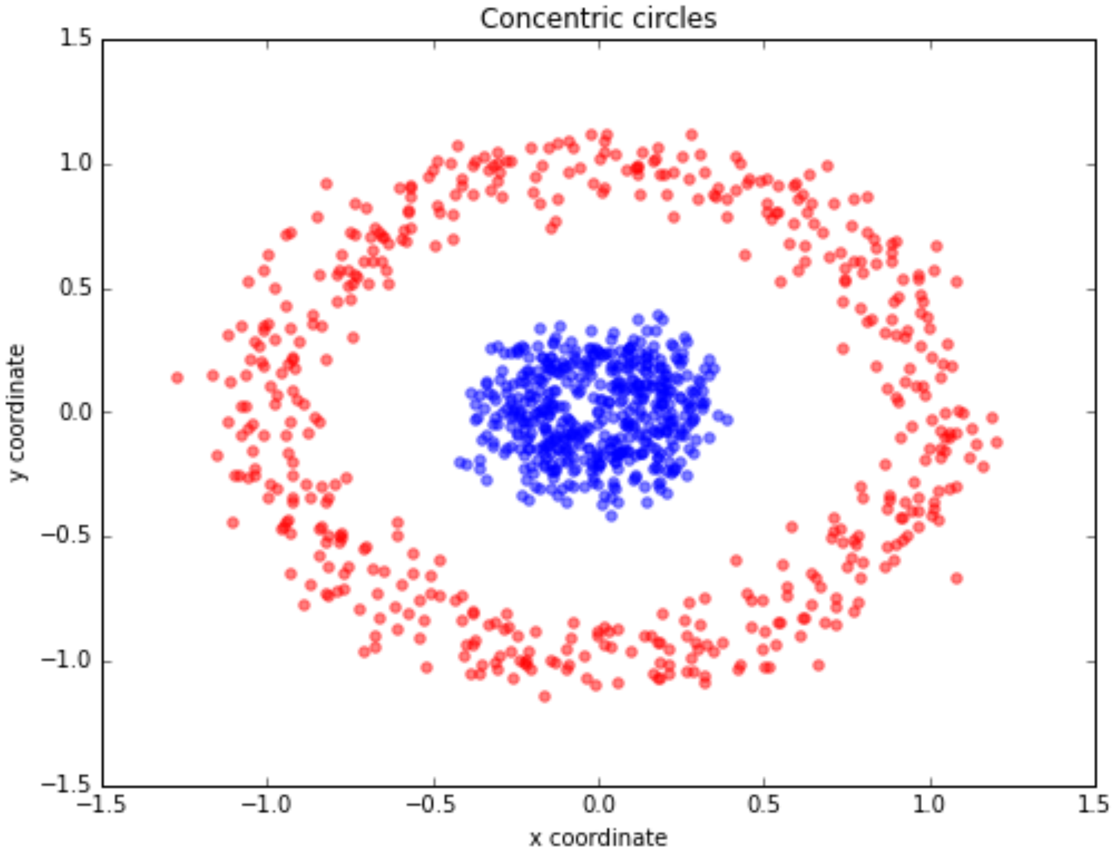


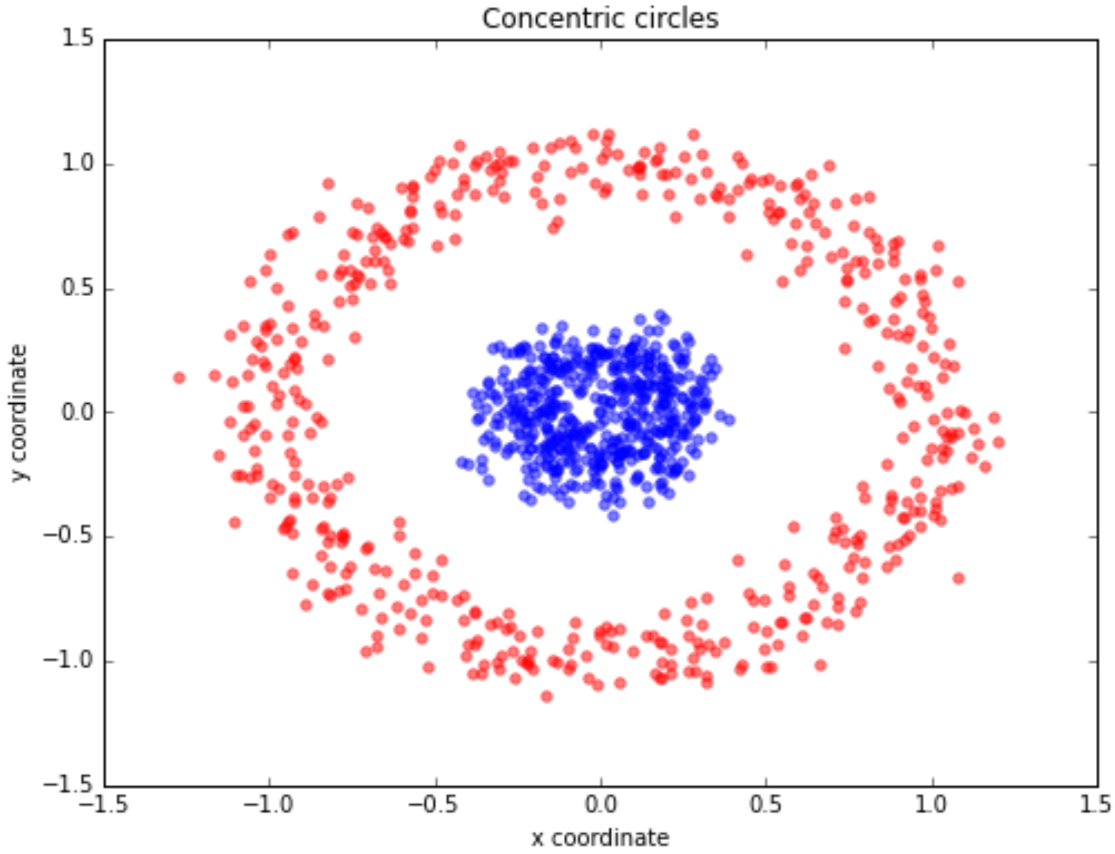
The idea is to construct a hyperplane that separates the data, making this a discriminative model

What if there is no easy hyperplane?

What if there is no easy hyperplane?

Walk with me on this,
a mathematical journey


Concentric circles
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y coordinate
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x coordinate



Pretty much no hyperplane will separate this out, but what if we could add a third dimension?

Q. OK fine, but what if I have 100 predictors? How many dimensions should I project into?

A. An arbitrary amount, possible infinite..

Q. OK fine, but what if I have 100 predictors? How many dimensions should I project into?

A. An arbitrary amount, possible infinite..

OK but this can take time..

Kernel Trick

We assume a certain shape of the data and the kernel trick saves us MASSIVE computation time

Kernel Trick

Example Gaussian or RBF (radial basis function)

$$
\begin{aligned}
K\left(x^{(i)}, x^{(j)}\right) &= \phi(x^{(i)})^T \phi(x^{(j)}) \\
&= \exp\left(-\gamma \left\|x^{(i)} - x^{(j)}\right\|^2\right), \qquad \gamma > 0
\end{aligned}
$$

Brings points out and makes a "pointed" manifold

Kernel Trick

Example:

Linear	(assumes a linear boundary)
Poly	(assumes a curved boundary)
Gaussian	(assumes a spherical boundary)

Pros

- Very fast training and predicting with kernel trick
- Built on solid mathematical foundation (unlike ANN)
- Very common and in sklearn
- It uses a small subset of training points in the decision function (the support vectors), so is memory efficient.

Cons

- A lot of “guess work” with kernels
- Hard to grasp math behind it (ok if you accept the black box)