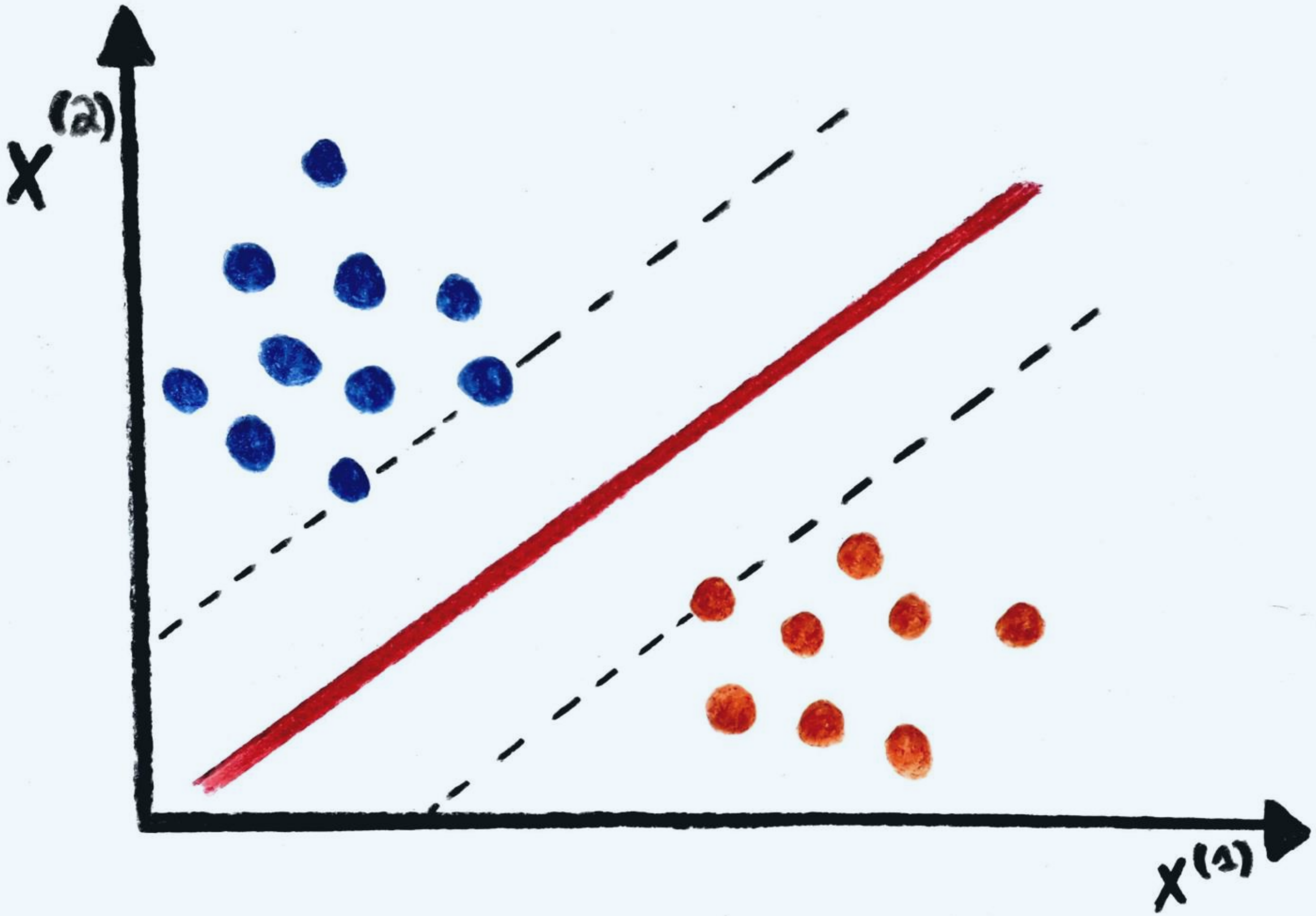
SUPPORT VECTOR MACHINES



Slides by Alberto Berni

LEARNING OUTCOMES

- The classification problem, bias & overfitting.

- Large Margin Classifiers.

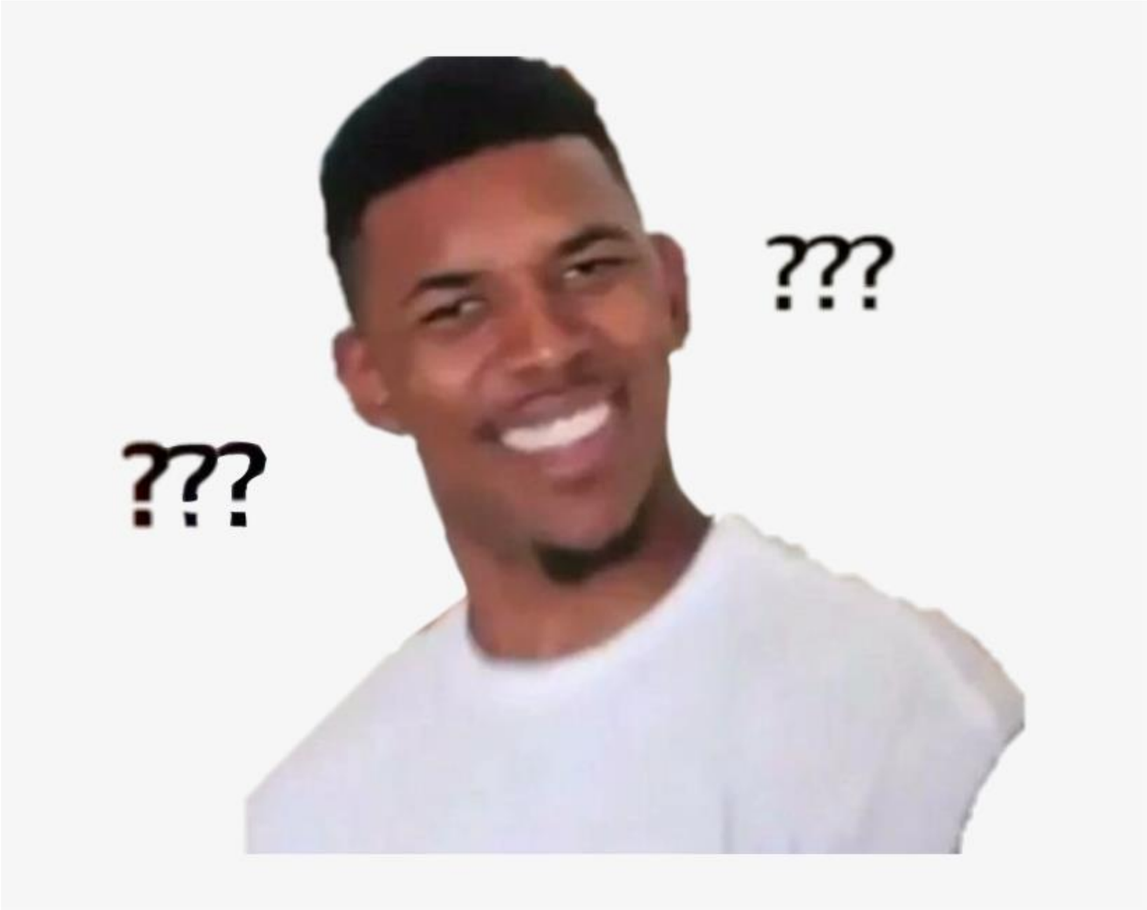
- Soft Margin Classifiers.

- Support Vector machines.

- Kernels for Support Vector Machines.

KAHOOT

TOOLKIT



TOOLKIT

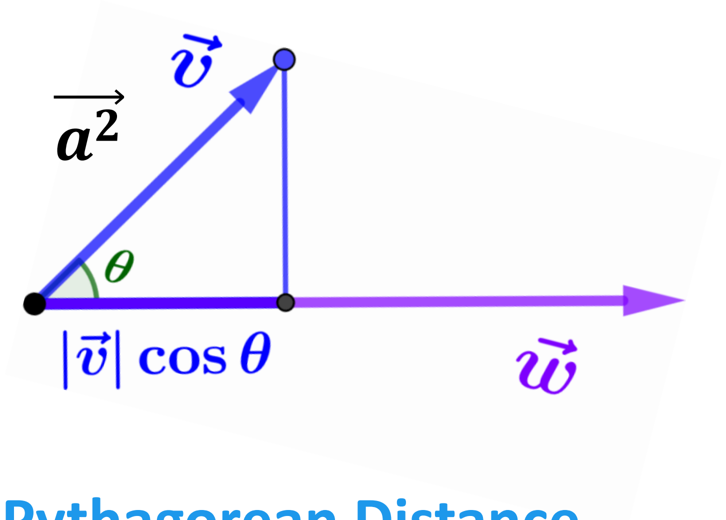
Dot Product

$$\vec{a} \cdot \vec{b}$$

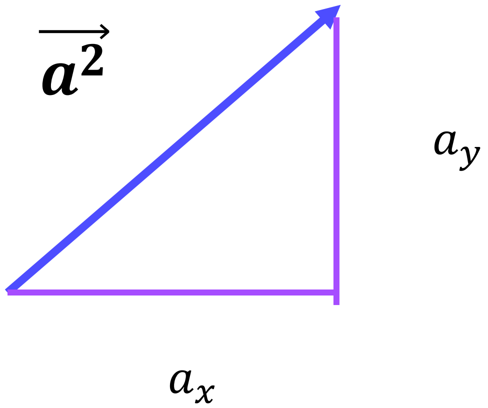
$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \bullet \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = a_x.b_x + a_y.b_y + a_z.b_z$$

$$\mathbf{u}.\mathbf{v} = \mathbf{u}^T\mathbf{v} = [u_1 u_2 \cdots u_n] \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = u_1v_1 + u_2v_2 + \cdots + u_nv_n$$

Vector Product

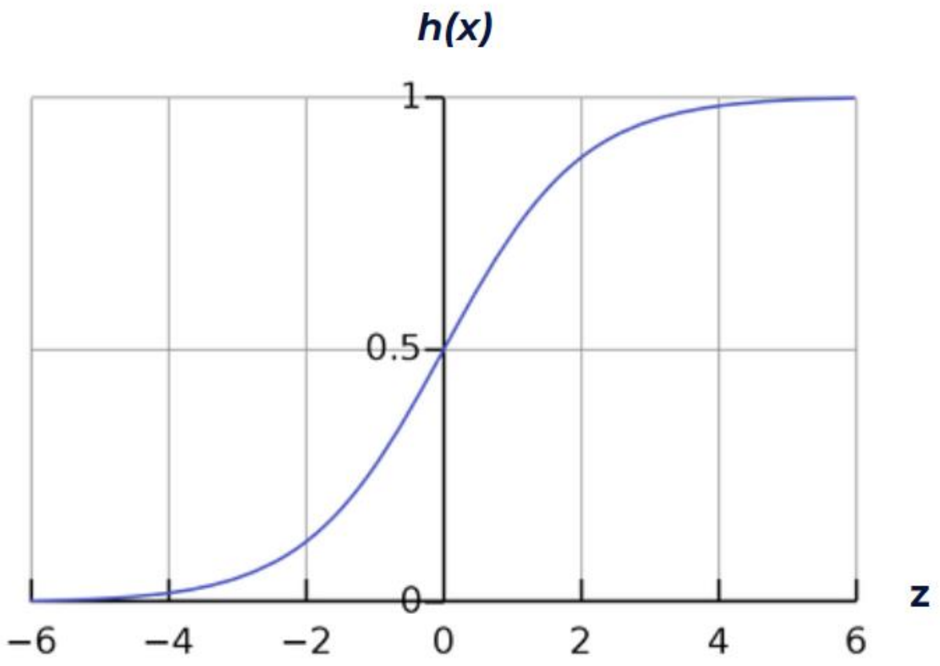


Pythagorean Distance



TOOLKIT

Logistic Regression



$$h(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$z = \theta^T x$$

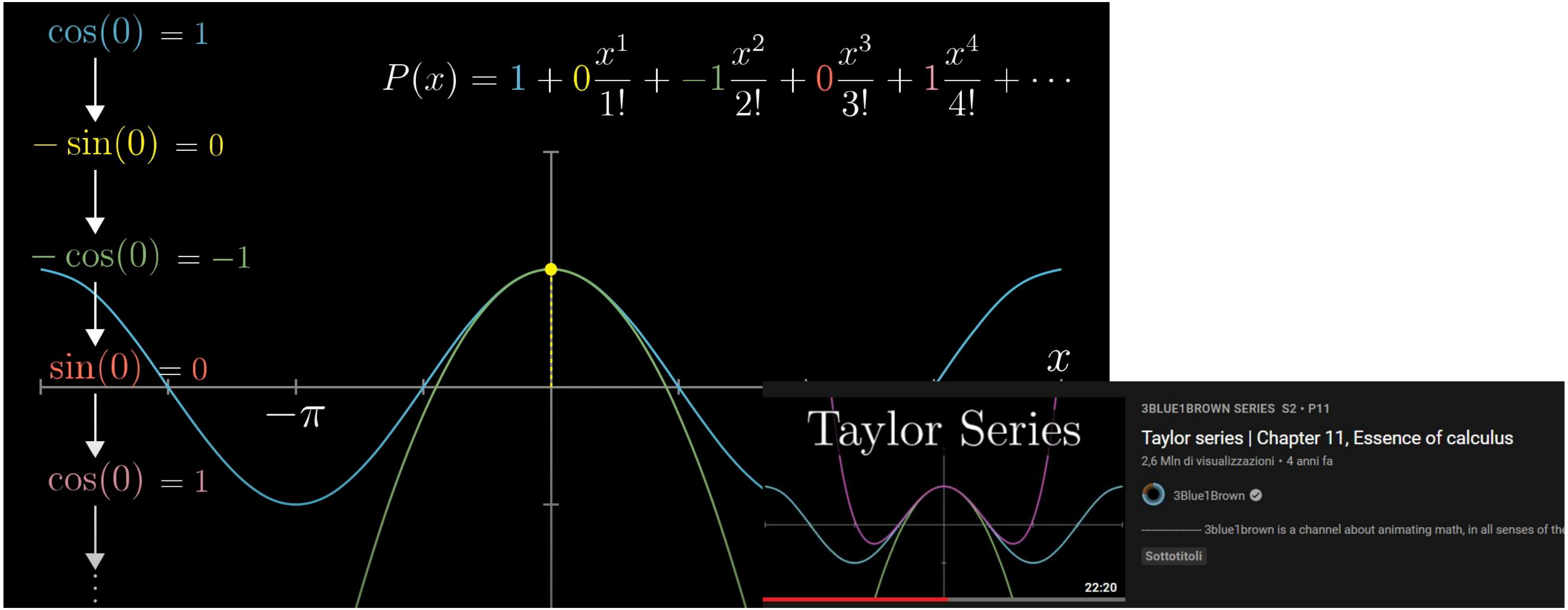
$$-y \log \frac{1}{1 + e^{-\theta^T x}} - (1 - y) \log(1 - \frac{1}{1 + e^{-\theta^T x}})$$

USAIS
University of Southampton AI Society

TOOLKIT

USAIS
University of Southampton AI Society

Taylor Series Expansions



TOOLKIT

USAIS
University of Southampton AI Society

Taylor Series Expansions

$$T_n(x) \approx f(x) \quad near \quad x = a$$

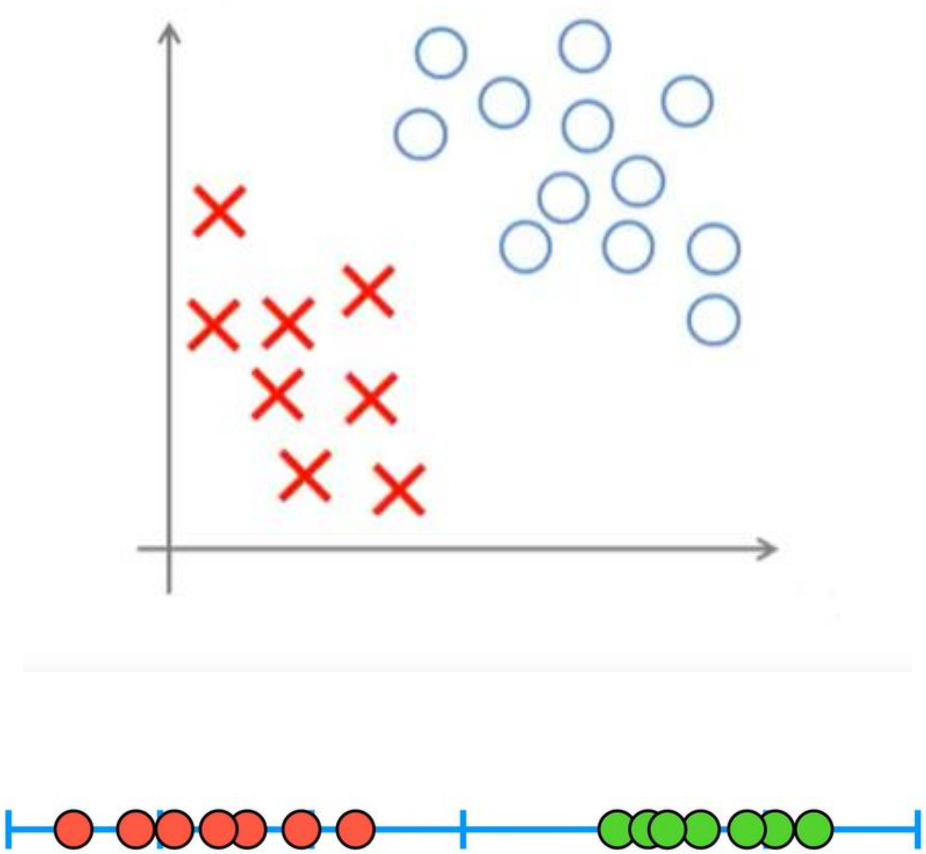
$$T_n(x) = c_0 + c_1(x-a) + c_2(x-a)^2 + \cdots + c_n(x-a)^n$$

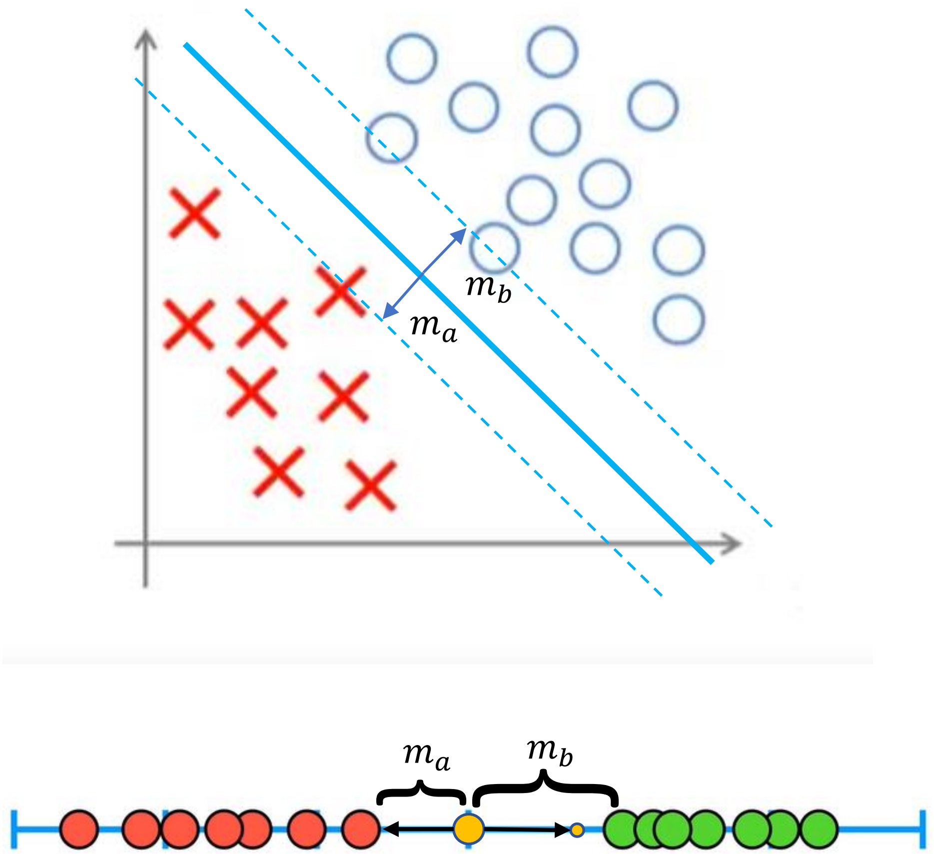
$$T_n(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f''(a)}{n!}(x-a)^n$$

LEARNING OUTCOMES

- The classification problem.
- Large Margin Classifiers.
- Support Vector machines.
- Kernels for Support Vector Machines.

THE CLASSIFICATION PROBLEM

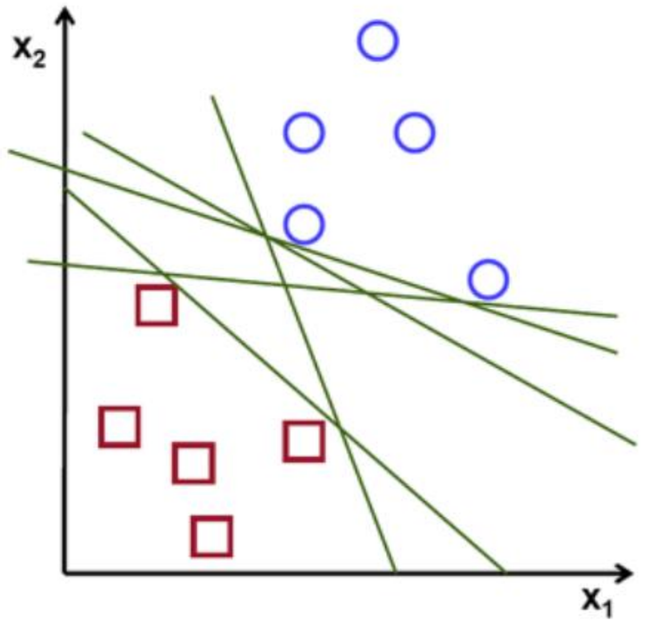


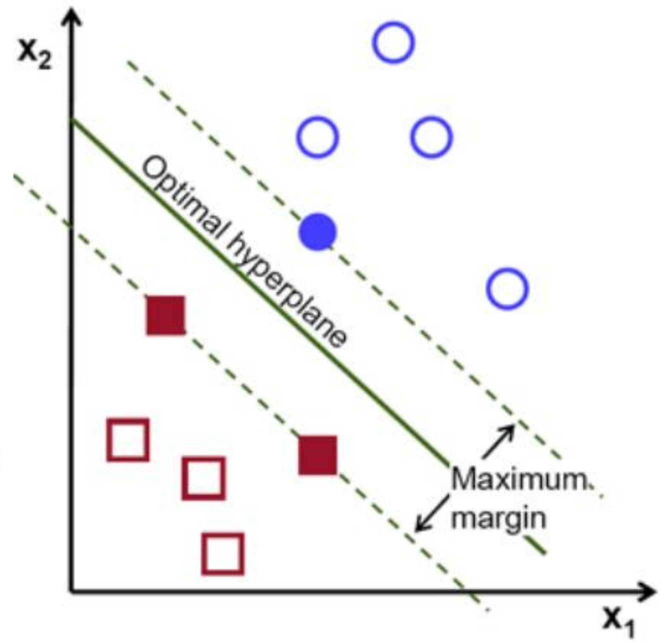


LEARNING OUTCOMES

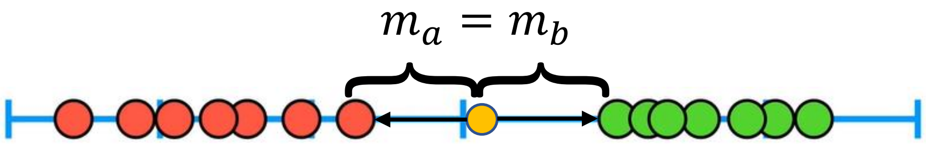
- The classification problem.
- Large Margin Classifiers.
- Support Vector machines.
- Kernels for Support Vector Machines.

MARGIN CLASSIFIERS



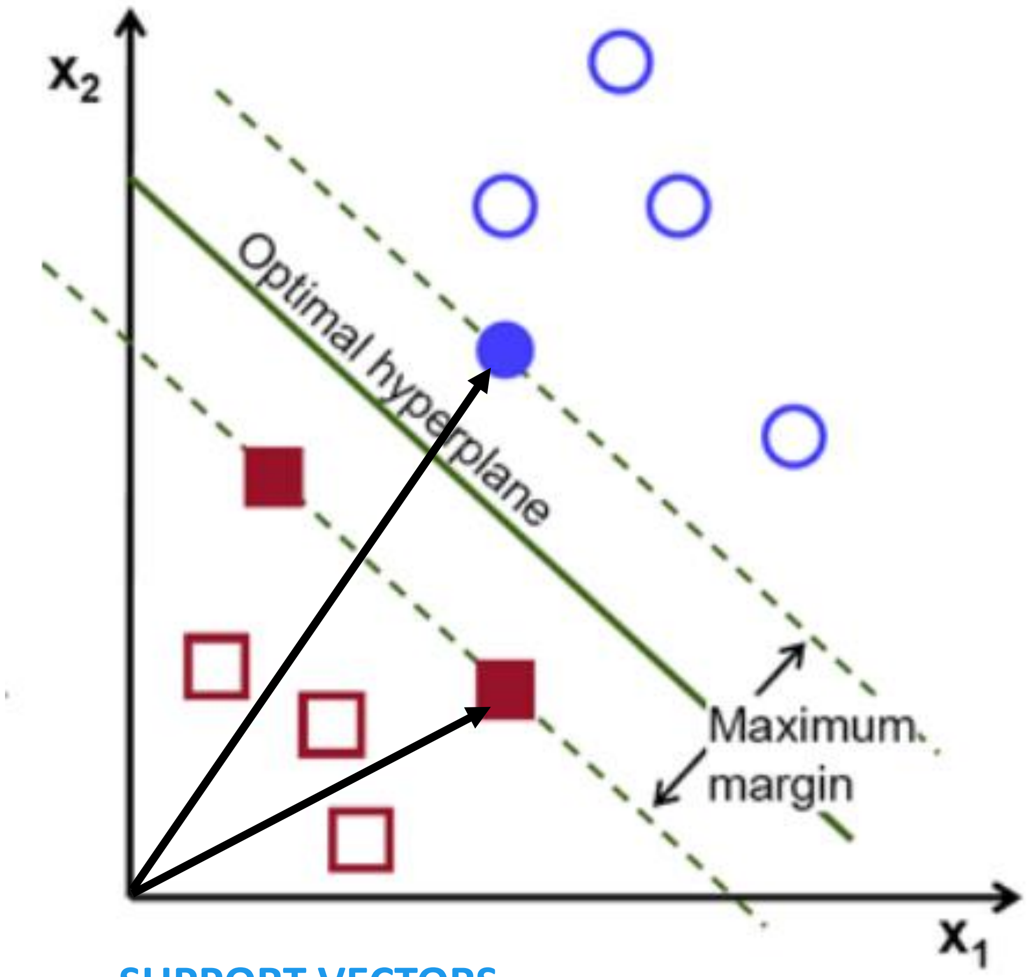


SMALL MARGIN CLASSIFICATION



MAXIMUM MARGIN CLASSIFICATION

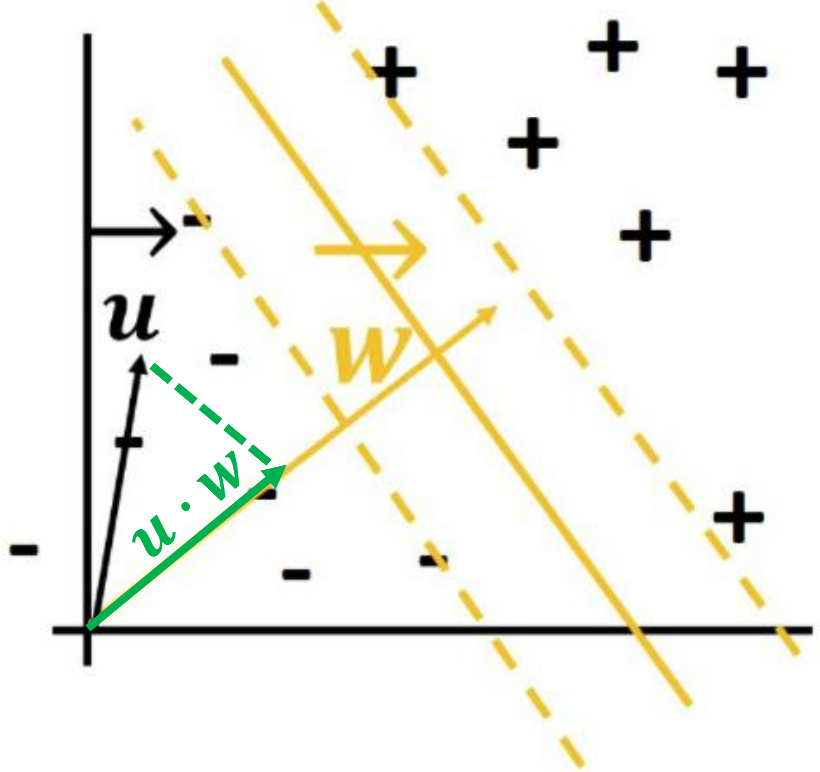
MARGIN CLASSIFIERS – MATH



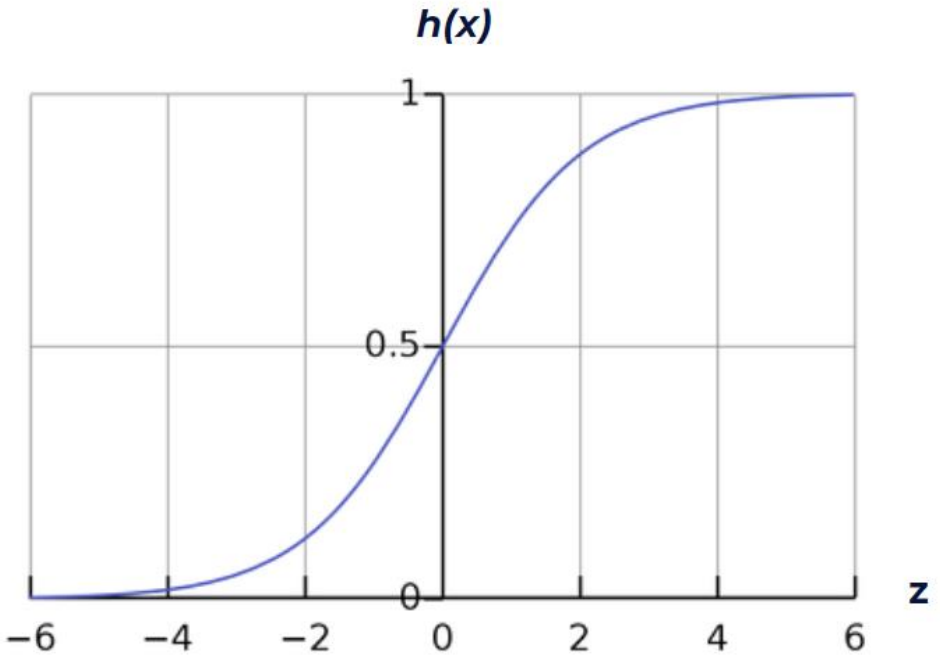
SUPPORT VECTORS

$\vec{w}$ Normal Vector

$\vec{u}$ Unknown Vector



MARGIN CLASSIFIERS – MATH



$$h(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$

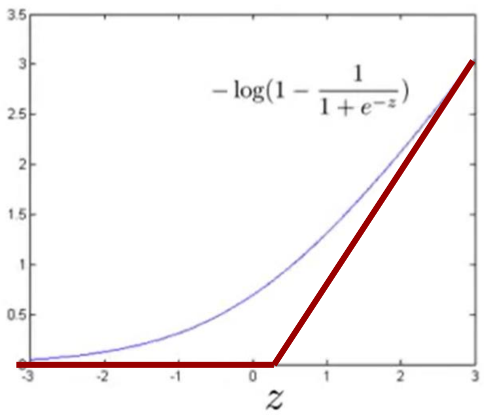
$$z = \theta^T x$$

Logistic Cost Function

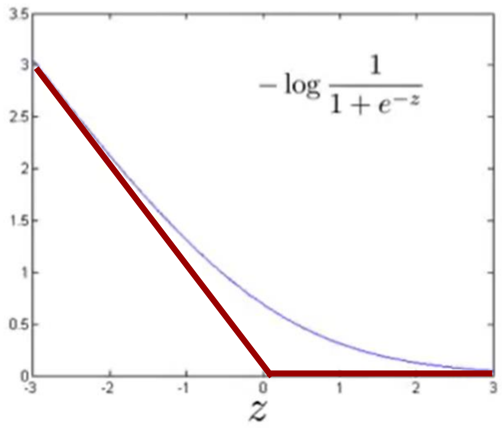
$$-y \log \frac{1}{1+e^{-\theta^T x}} - (1-y)\log(1 - \frac{1}{1+e^{-\theta^T x}})$$

when y = 1 **when y = 0**

If $y = 0$ (want $\theta^T x \ll 0$):



If $y = 1$ (want $\theta^T x \gg 0$):



MARGIN CLASSIFIERS – MATH

USAIS
University of Southampton AI Society

Logistic Cost Function

$$-y \log \frac{1}{1+e^{-\theta^T x}} - (1-y)\log(1 - \frac{1}{1+e^{-\theta^T x}})$$

$cost_1$ $\quad$ $cost_0$

$$\log \frac{1}{1+e^{-\theta^T x}} \qquad \log(1 - \frac{1}{1+e^{-\theta^T x}})$$

when y = 1

when y = 0

$$C = \frac{1}{\lambda}$$

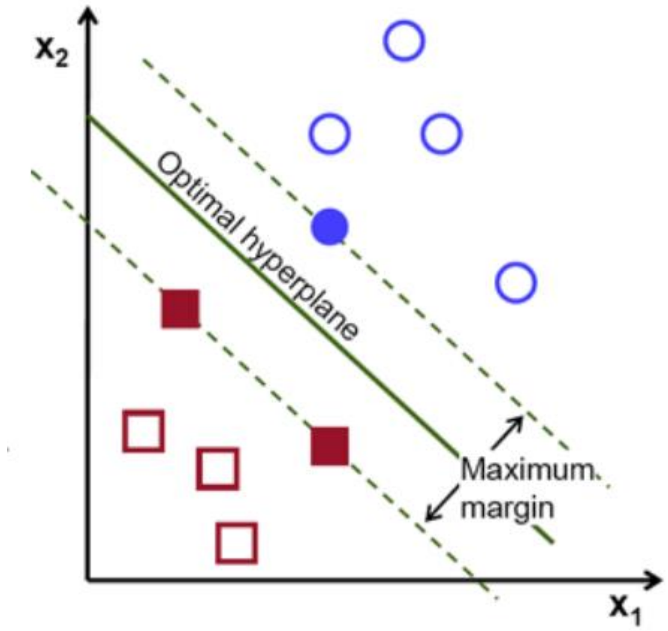
$$\min_\theta C \sum_{i=1}^{m} \left[0^{(i)} \cancel{cost_1(\theta^T x^{(i)})} + (1 - 1^{(i)}) \cancel{cost_0(\theta^T x^{(i)})} \right] + \frac{1}{2}\sum_{i=1}^{n} \theta_j^2$$

LEARNING OUTCOMES

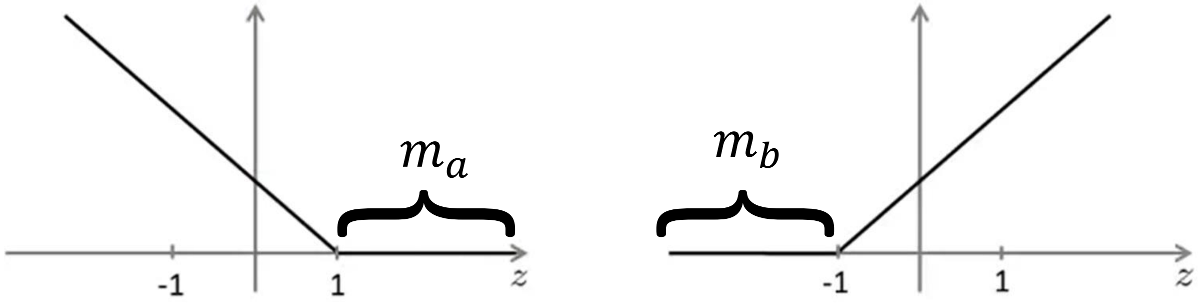
- The classification problem.
- Large Margin Classifiers.
- Support Vector machines.
- Kernels for Support Vector Machines.

MARGIN CLASSIFIERS – MATH

$$-y \log \frac{1}{1+e^{-\theta^T x}} - (1-y)\log(1 - \frac{1}{1+e^{-\theta^T x}})$$



$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$



$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

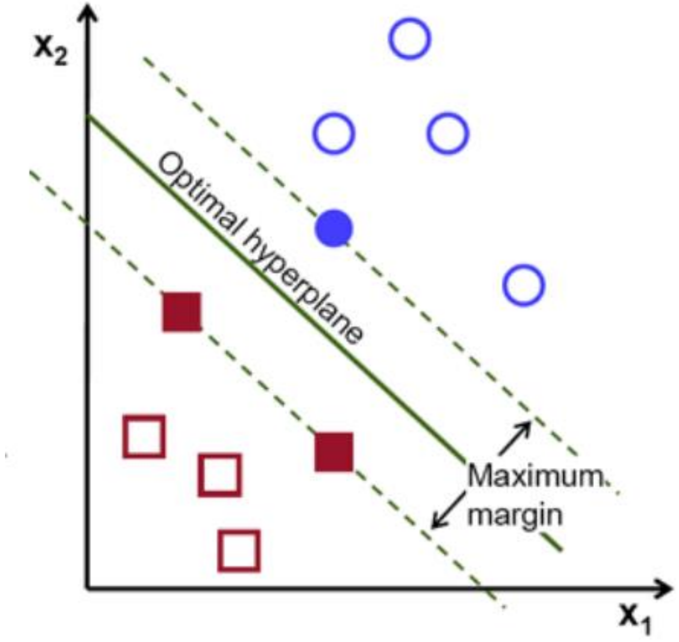
$$\begin{aligned} \text{s.t.} \quad & \theta^T x^{(i)} \geq 1 && \text{if } y^{(i)} = 1 \\ & \theta^T x^{(i)} \leq -1 && \text{if } y^{(i)} = 0 \end{aligned}$$

KERNELS

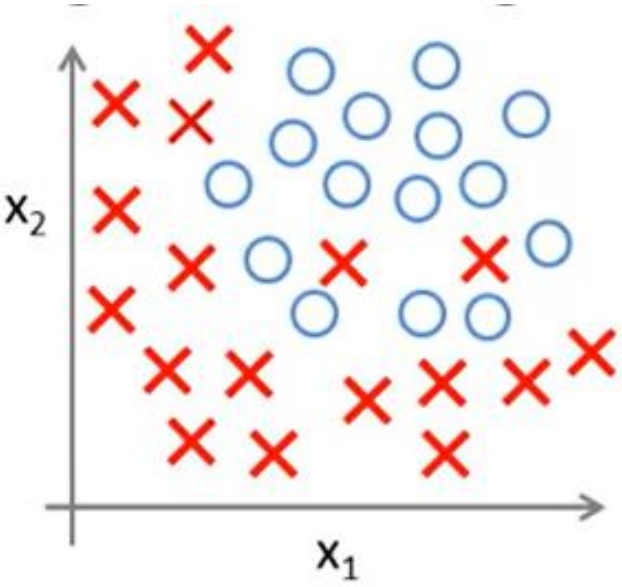
USAIS
University of Southampton AI Society

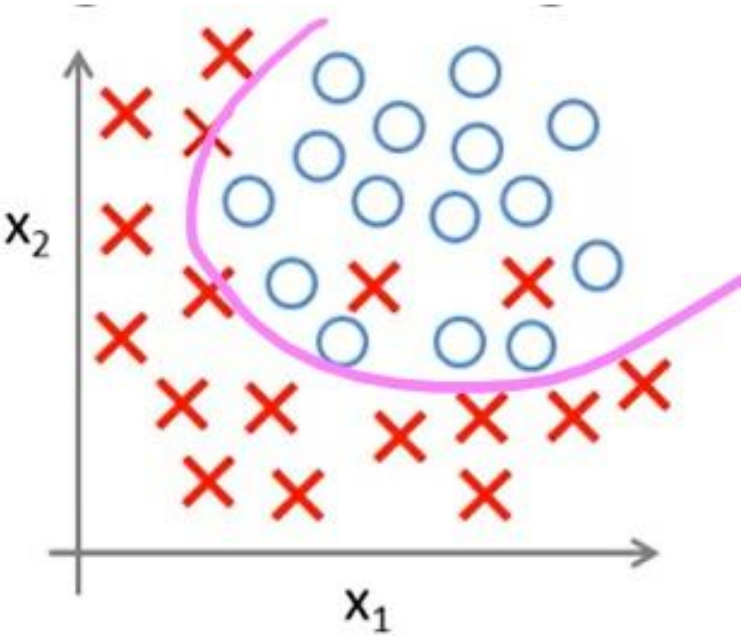
Kernels

a support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space.



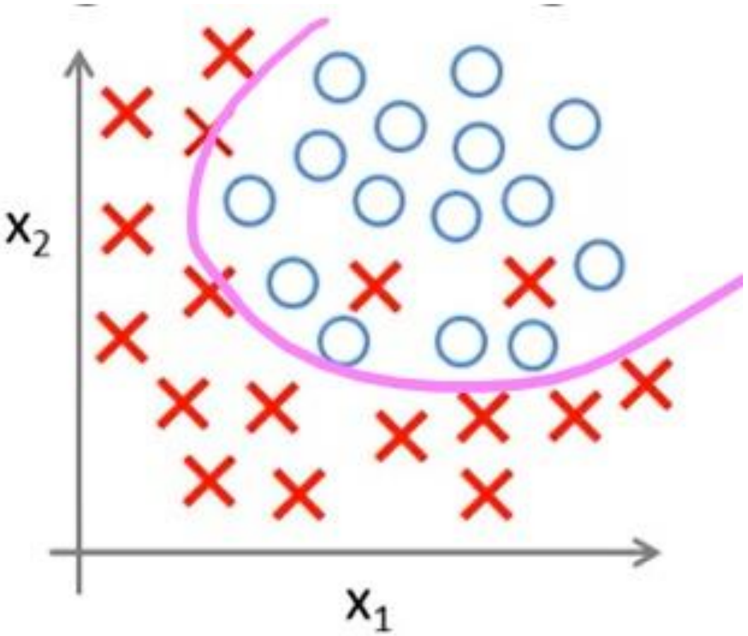
KERNELS

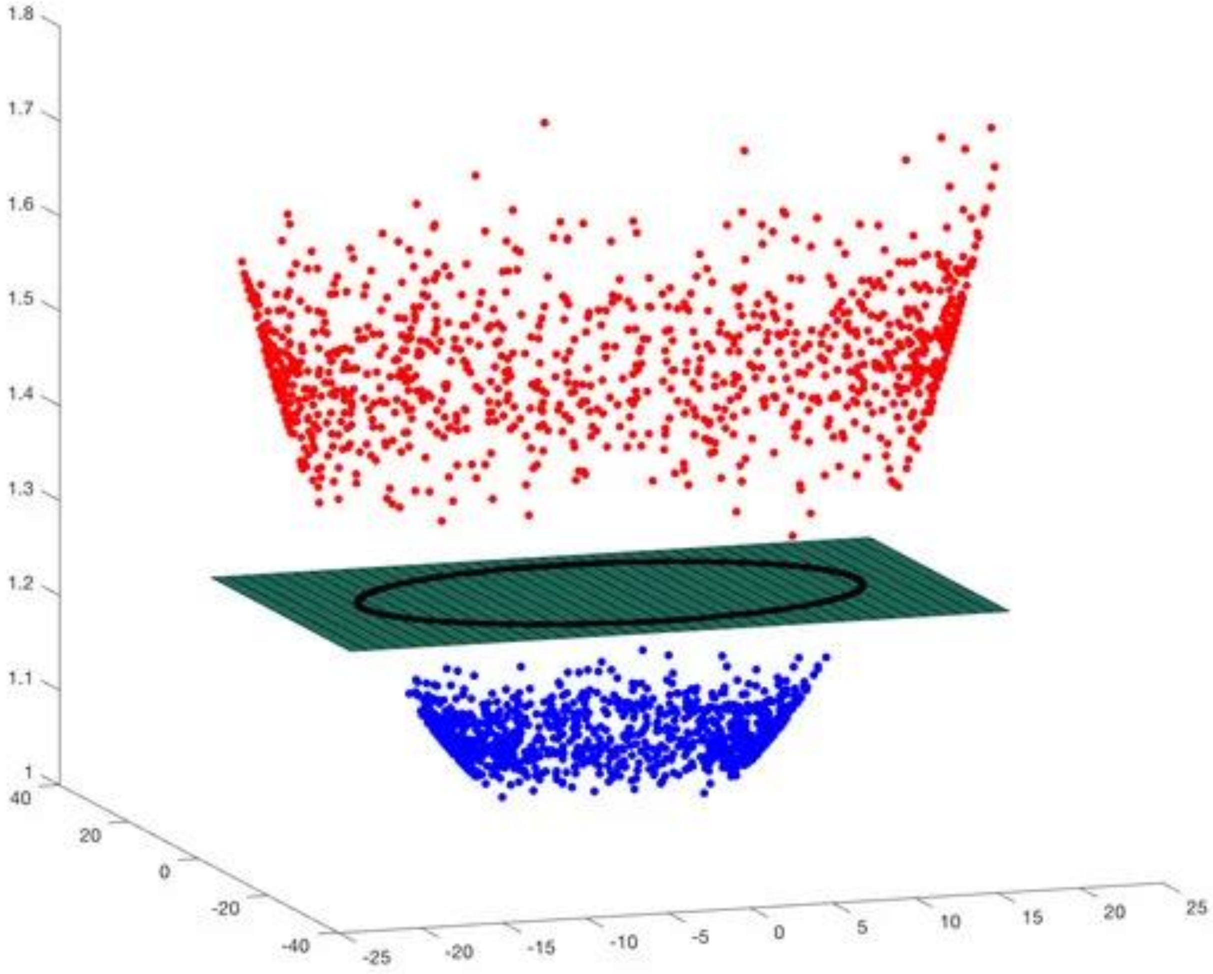




KERNELS

USAIS
University of Southampton AI Society





LEARNING OUTCOMES

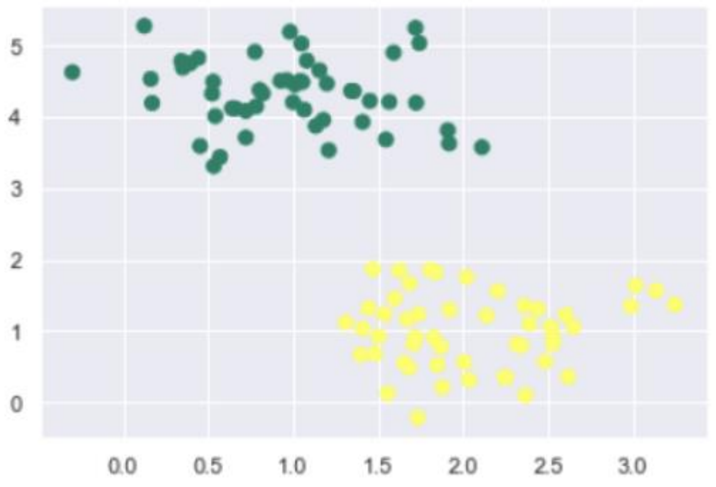
- The classification problem.
- Large Margin Classifiers.
- Support Vector machines.
- Kernels for Support Vector Machines.

SMOL BREK

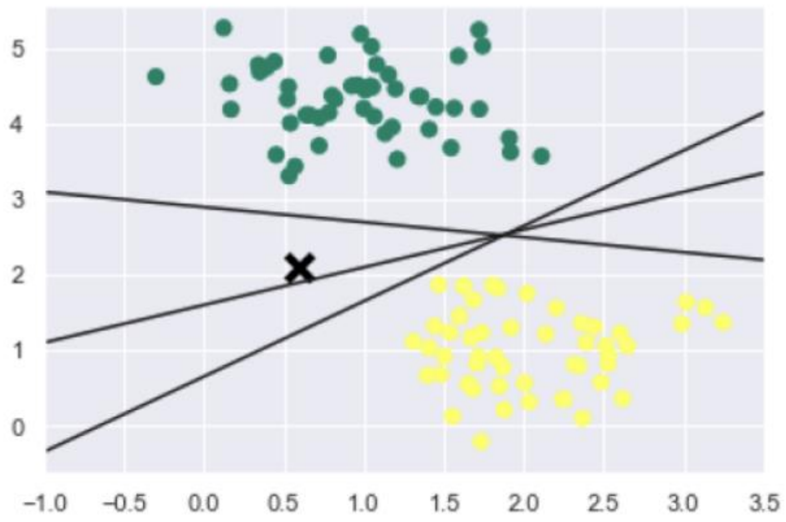
MARGIN CLASSIFIERS

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns; sns.set()
```

```
from sklearn.datasets.samples_generator import make_blobs
X, y = make_blobs(n_samples=100, centers=2, random_state=0, cluster_std=0.50)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer');
```

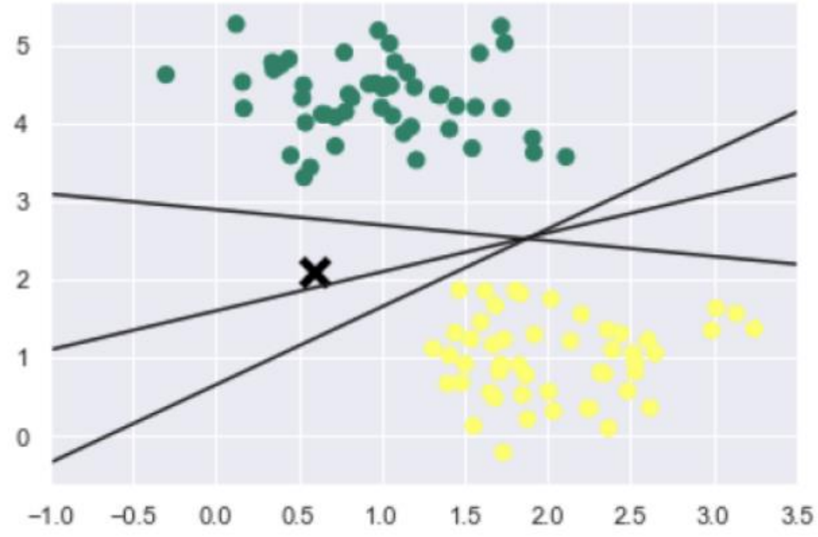



```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer')
plt.plot([0.6], [2.1], 'x', color='black', markeredgewidth=4, markersize=12)
for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:
   plt.plot(xfit, m * xfit + b, '-k')
plt.xlim(-1, 3.5);
```



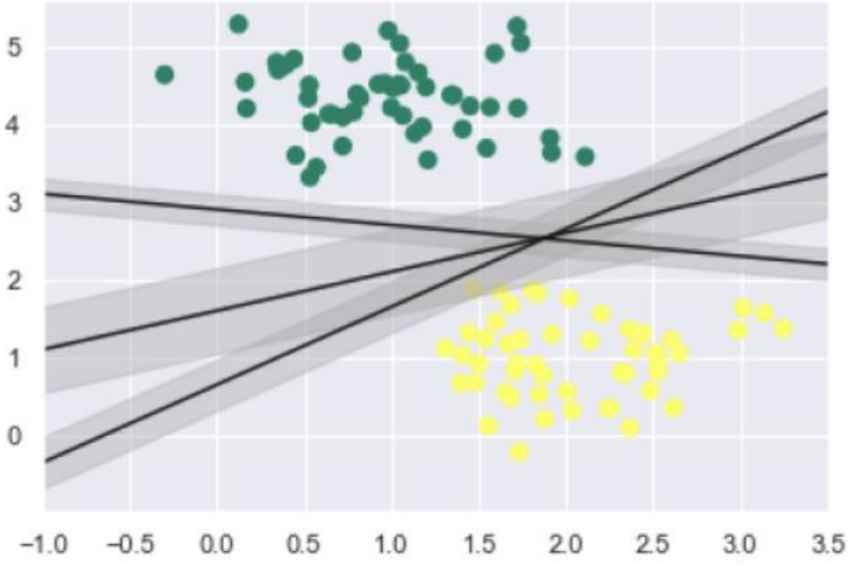
MARGIN CLASSIFIERS

```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer')
   for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
   yfit = m * xfit + b
   plt.plot(xfit, yfit, '-k')
   plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
         color='#AAAAAA', alpha=0.4)
plt.xlim(-1, 3.5);
```




```
from sklearn.svm import SVC # "Support vector classifier"
model = SVC(kernel='linear', C=1E10)
model.fit(X, y)
```

```
SVC(C=10000000000.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
kernel='linear', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001, verbose=False)
```



MARGIN CLASSIFIERS

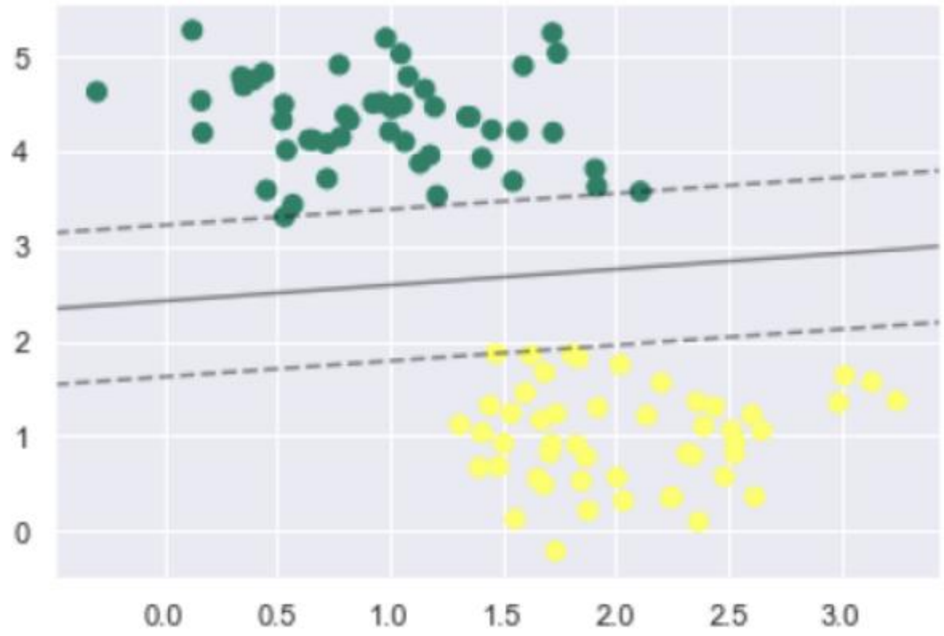
```python
def decision_function(model, ax=None, plot_support=True):
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
```

```python
x = np.linspace(xlim[0], xlim[1], 30)
y = np.linspace(ylim[0], ylim[1], 30)
Y, X = np.meshgrid(y, x)
xy = np.vstack([X.ravel(), Y.ravel()]).T
P = model.decision_function(xy).reshape(X.shape)
```

```python
ax.contour(X, Y, P, colors='k',
    levels=[-1, 0, 1], alpha=0.5,
    linestyles=['--', '-', '--'])
```

```python
if plot_support:
    ax.scatter(model.support_vectors_[:, 0],
        model.support_vectors_[:, 1],
        s=300, linewidth=1, facecolors='none');
ax.set_xlim(xlim)
ax.set_ylim(ylim)
```

```python
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer')
decision_function(model);
```



Polynomial Kernel Math

KERNELS

Intro to n-dimensional approach (spiral kernel)

KERNELS

n-dimensional approach

(spiral kernel) maths

SMOL BREK

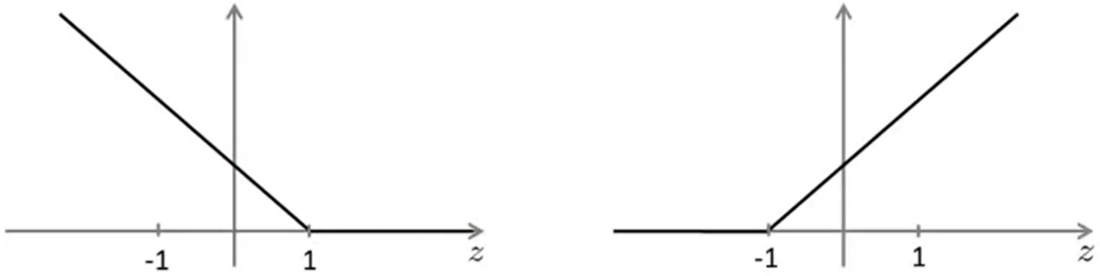
CODE

CODE GOES HERE

Mathematical Intuition

Support Vector Machine

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$



If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)

If $y = 0$, we want $\theta^T x \leq -1$ (not just < 0)

- Introduce cost function derivation from Log reg. equation

Math int 2

(x,y)

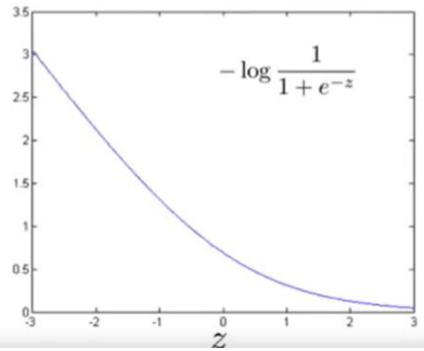
Alternative view of logistic regression

Cost of example: $-(y \log h_\theta(x) + (1-y)\log(1-h_\theta(x)))$

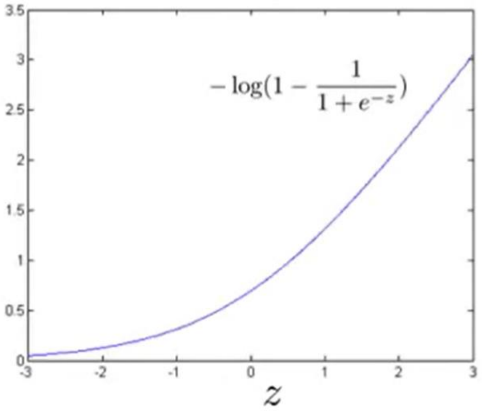
$$= -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y)\log(1 - \frac{1}{1+e^{-\theta^T x}})$$

If $y = 1$ (want $\theta^T x \gg 0$):

If $y = 1$ (want $\theta^T x \gg 0$):



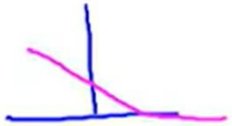
If $y = 0$ (want $\theta^T x \ll 0$):



Math int 3

Support vector machine

Logistic regression:



$$\min_\theta \underbrace{\frac{1}{m}\left[\sum_{i=1}^m y^{(i)}\underbrace{\left(-\log h_\theta(x^{(i)})\right)}_{cost_1(\theta^T x^{(i)})} + (1-y^{(i)})\underbrace{\left((-\log(1-h_\theta(x^{(i)})))\right)}_{cost_0(\theta^T x^{(i)})}\right]}_{A} + \underbrace{\frac{\lambda}{2m}\sum_{j=1}^n\theta_j^2}_{B}$$

Support vector machine:

$$\min_\theta \cancel{\frac{1}{m}}\; C \sum_{i=1}^m y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)})\, cost_0(\theta^T x^{(i)}) + \frac{1\cancel{\lambda}}{2\cancel{m}}\sum_{i=0}^n \theta_j^2$$

$$\min_u \left((u-5)^2+1\right) \rightarrow u=5 \qquad \times 10$$

$$\min_u 10(u-5)^2+10 \rightarrow u=5$$

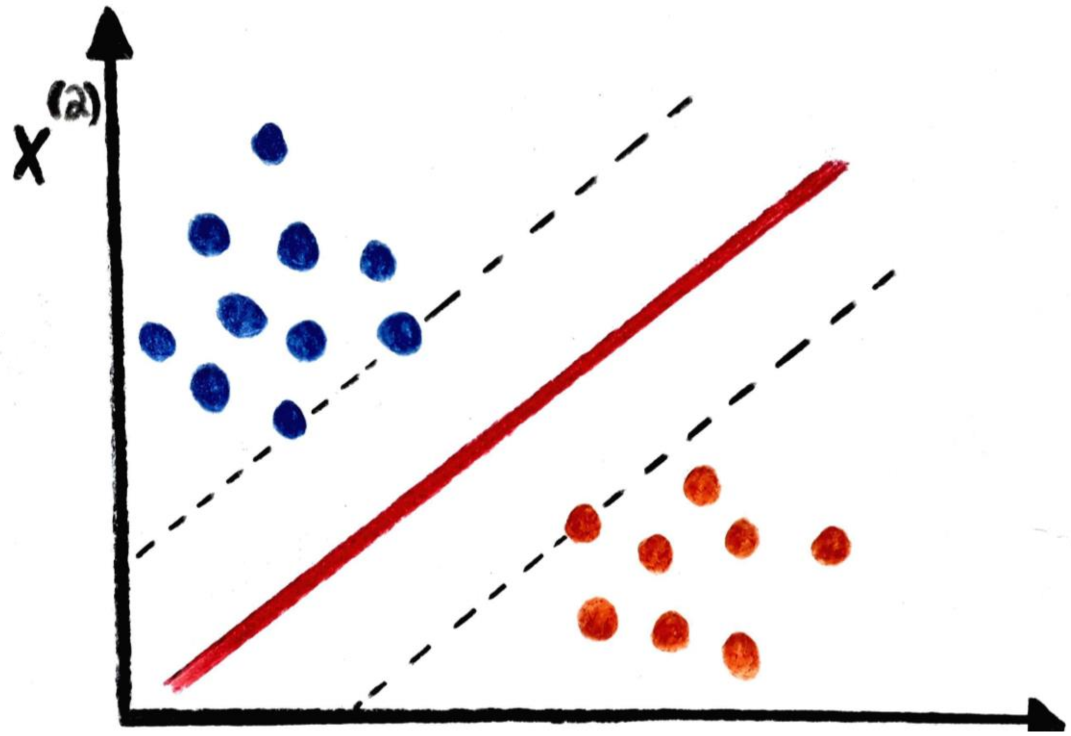
$$A + \lambda B$$

$$C A + B$$

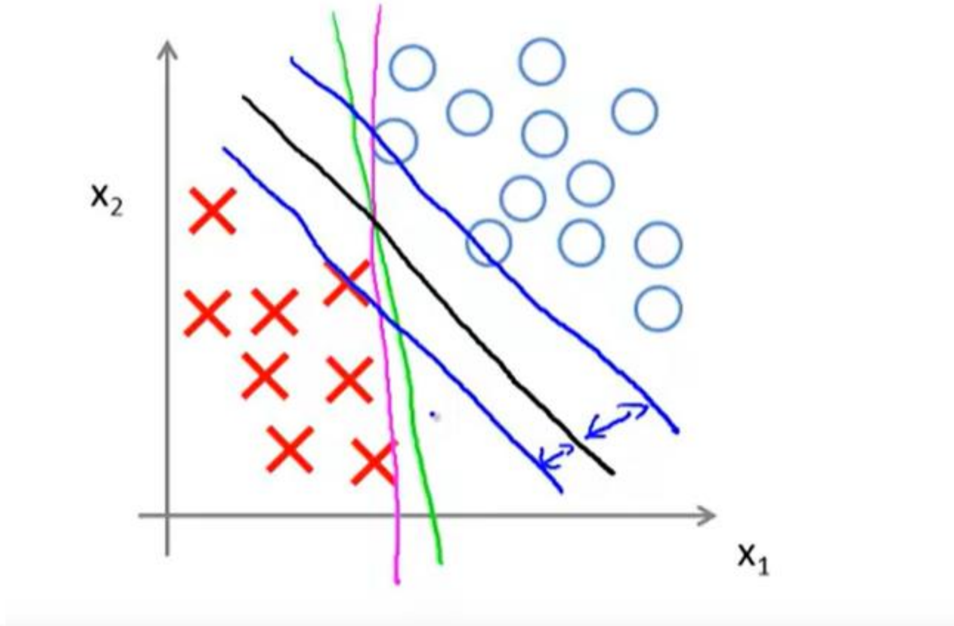
$$C = \frac{1}{\lambda}$$

$$\min_\theta C\sum_{i=1}^m\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^n\theta_j^2$$

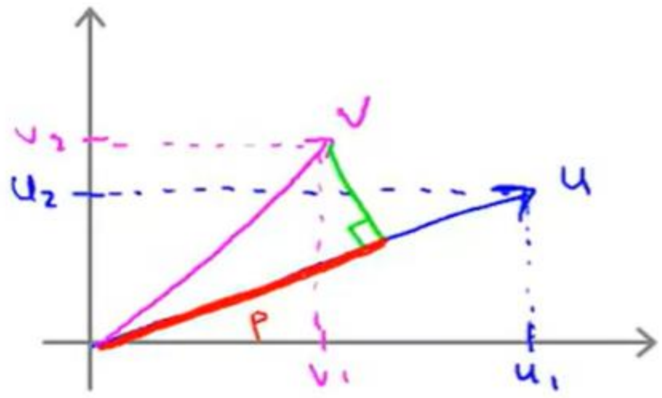
Issues with LMI

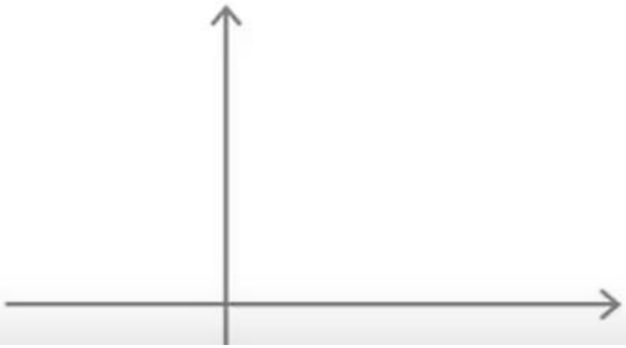


SVM Decision Boundary: Linearly separable case



Vector Inner Product





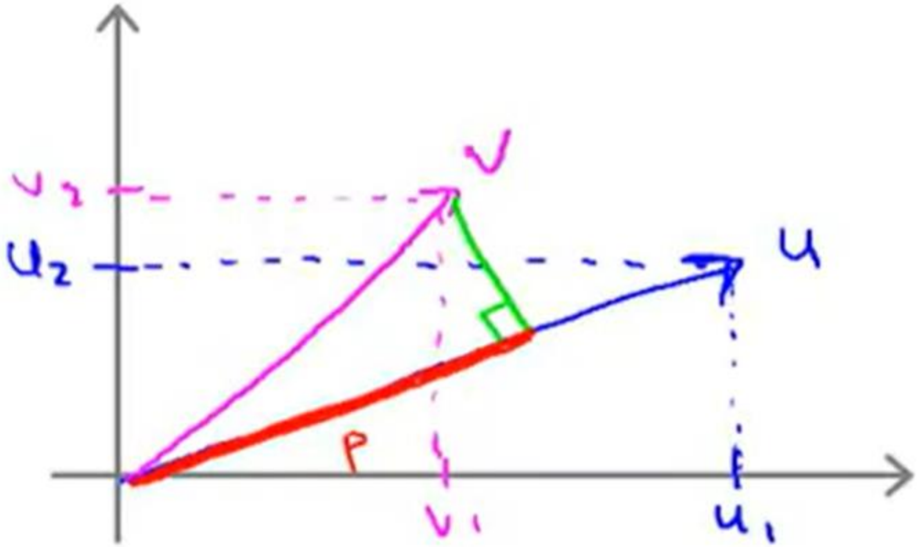
$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

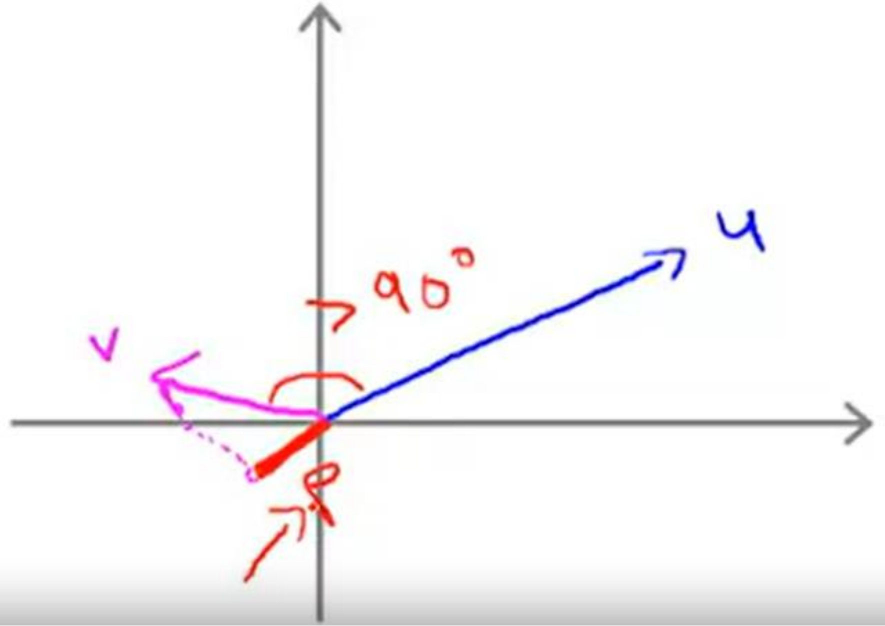
$u^T v = ?$

$\|u\|$ = length of vector u

$= \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$

Vector Inner Product





$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$u^T v = ?$ $\qquad [u_1 \;\; u_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$

$\|u\|$ = length of vector u

$= \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$

p = length of projection of v onto u.

signed

$u^T v = \underline{p} \cdot \underline{\|u\|}$ $\qquad = v^T u$

$= u_1 v_1 + u_2 v_2$ $\qquad p \in \mathbb{R}$

$u^T v = p \cdot \|u\|$

$p < 0$

Andrew N

$\omega = (\sqrt{\omega})^2$

SVM Decision Boundary

$$\min_\theta \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \frac{1}{2}(\theta_1^2+\theta_2^2) = \frac{1}{2}\left(\underbrace{\sqrt{\theta_1^2+\theta_2^2}}_{=\|\theta\|}\right)^2 = \frac{1}{2}\|\theta\|^2$$

$$\text{s.t.}\quad \boxed{\theta^T x^{(i)} \geq 1} \quad \text{if } y^{(i)} = 1$$

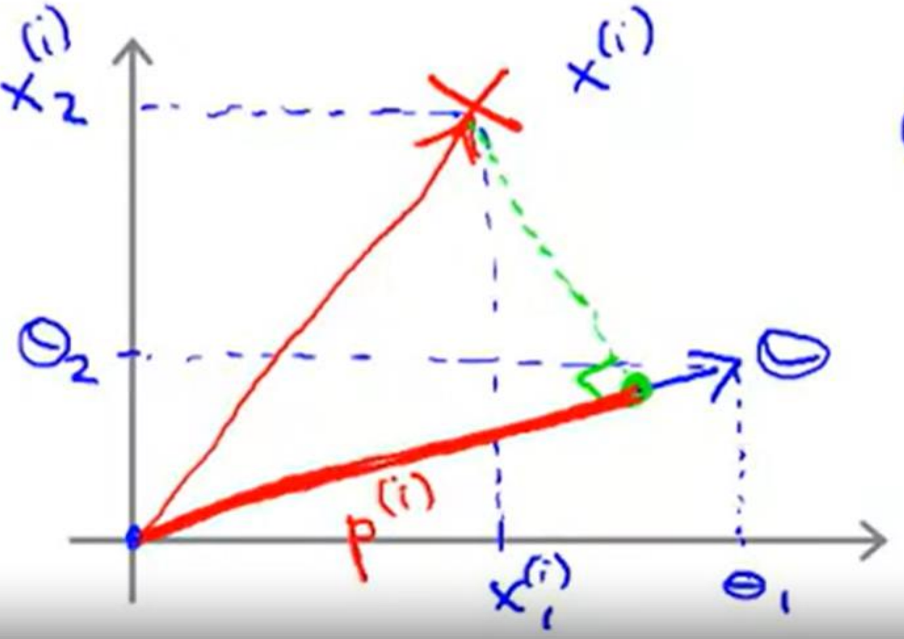
$$\theta^T x^{(i)} \leq -1 \quad \text{if } y^{(i)} = 0$$

Simplication: $\theta_0 = 0$. $\quad n=2$

$$\begin{bmatrix} \cancel{\theta_0} \\ \theta_1 \\ \theta_2 \end{bmatrix} \quad \theta_0 = 0$$

$\theta^T x^{(i)} = ?$

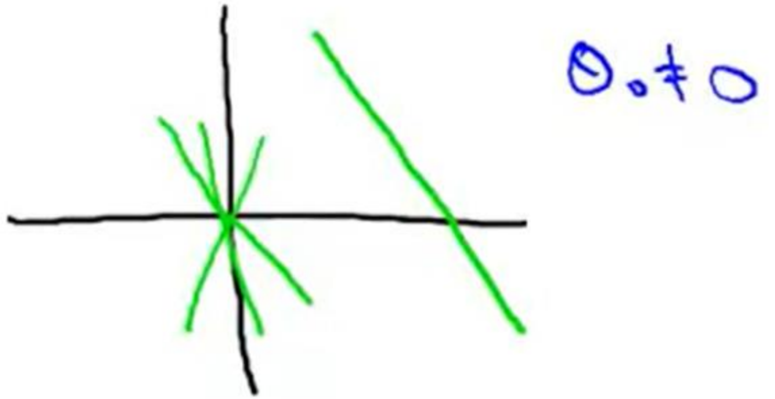
$u^T v$



$$\theta^T x^{(i)} = \boxed{p^{(i)} \cdot \|\theta\|}$$

$$= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}$$

SVM Decision Boundary



$$\min_\theta \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \frac{1}{2}\|\theta\|^2$$

s.t. $\boxed{p^{(i)} \cdot \|\theta\| \geq 1}$ if $y^{(i)} = 1$

$p^{(i)} \cdot \|\theta\| \leq -1$ if $y^{(i)} = 1$

C very large

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplification: $\theta_0 = 0$