Master in Internet of Things for eHealth

M5. Smart Data Knowledge / Analytics

Support Vector Machines

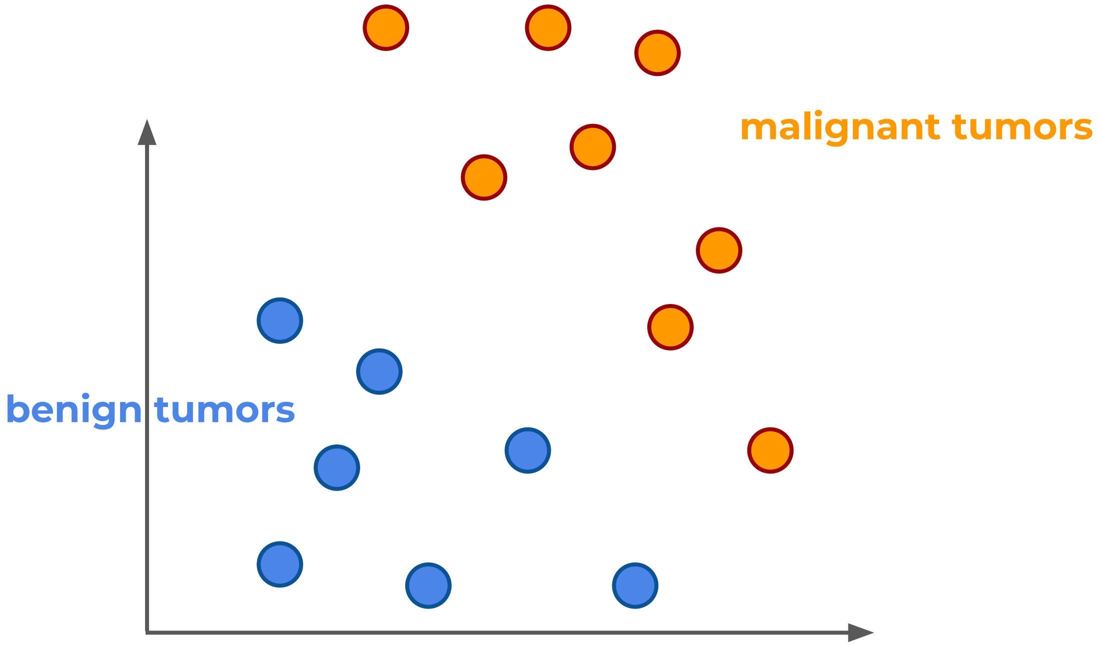
Instructor **David Gerónimo**
research@davidgeronimo.com

Universitat Autònoma de Barcelona

January 9th 2019

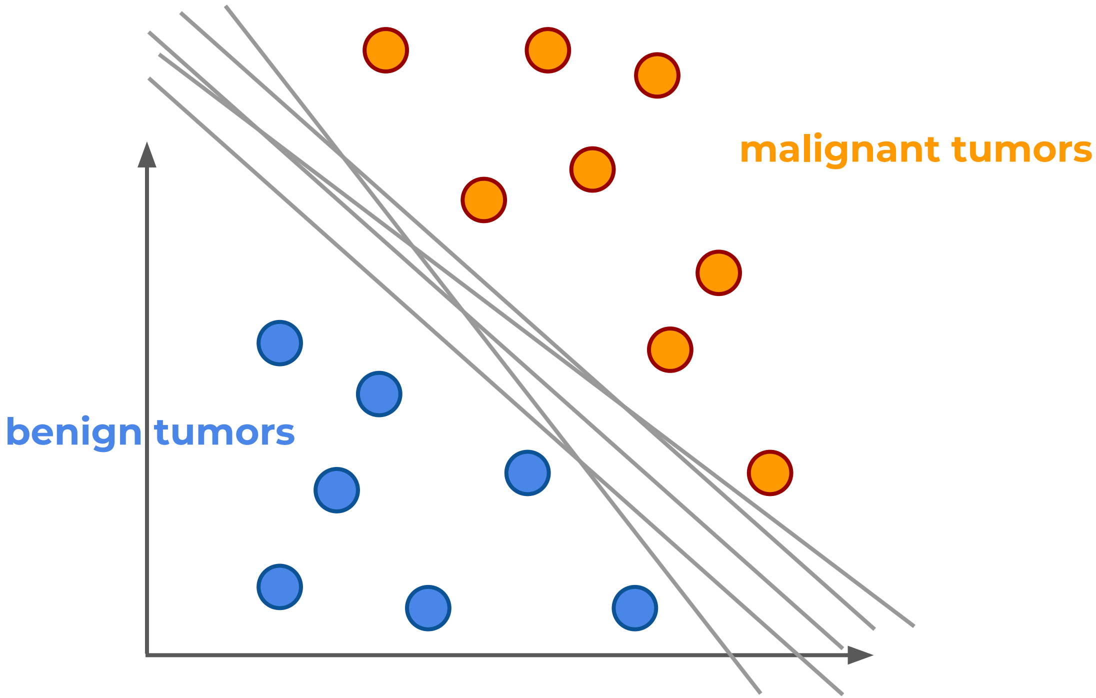
Support Vector Machines

- **What is the optimal way of separating a set of points in a space?**



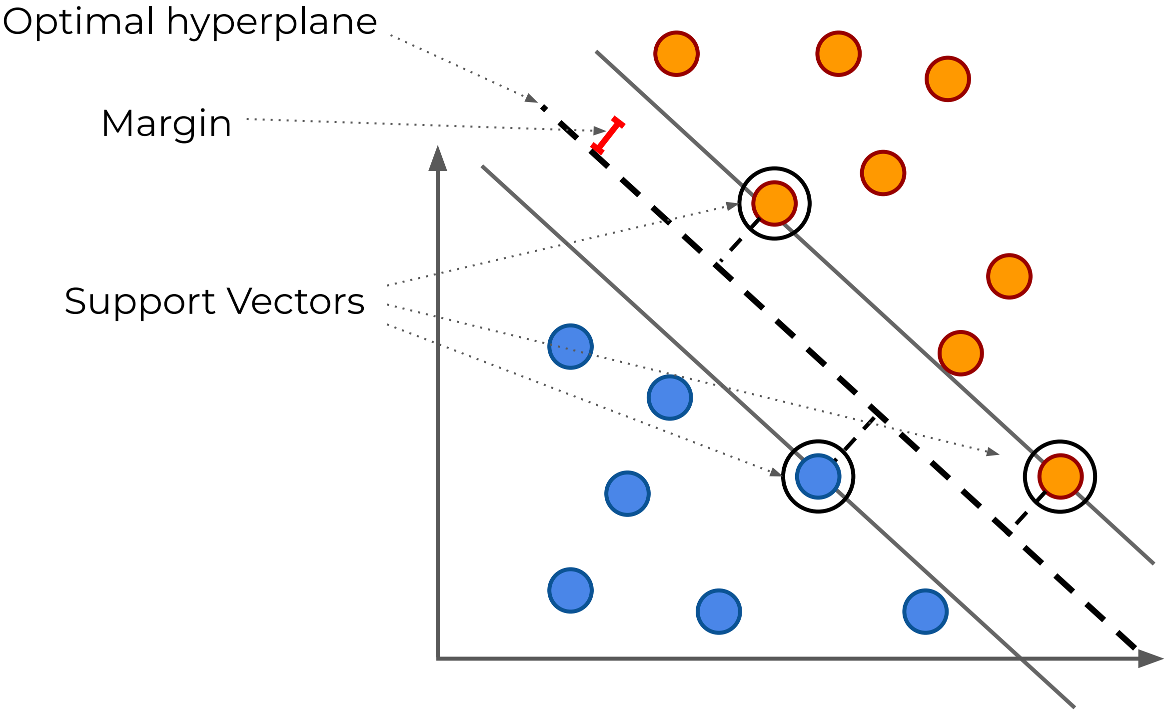
Support Vector Machines

- What is the optimal way of separating a set of points in a space?



Support Vector Machines

- **What is the optimal way of separating a set of points in a space?**



Vladimir Vapnik (1990s)

“The one with the largest margin”

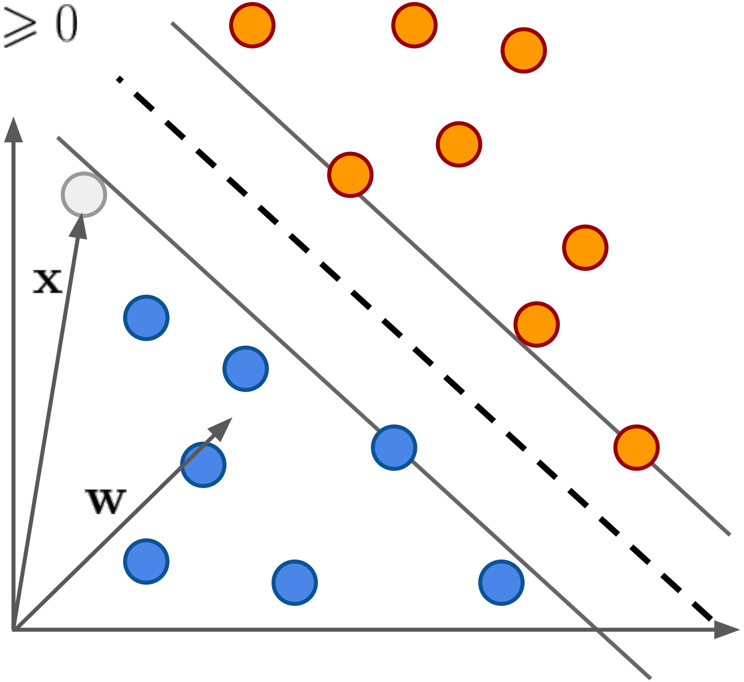
Support Vector Machines

- **Let's formalize it!**

If positive = malignant

decision rule: $\mathbf{w}^{\mathbf{T}}\mathbf{x} + b \geqslant 0$

If negative = benignant



Support Vector Machines

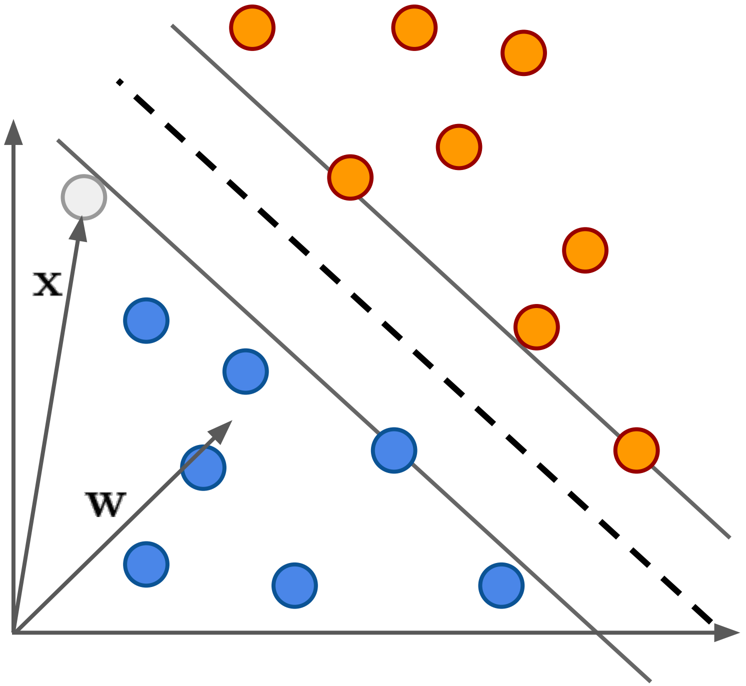
- **But we want a margin**

If positive = malignant

$$\mathbf{w}^{\mathbf{T}}\mathbf{x} + b \geqslant +1$$

If negative = benignant

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



Support Vector Machines

- **We simplify the equations for convenience**

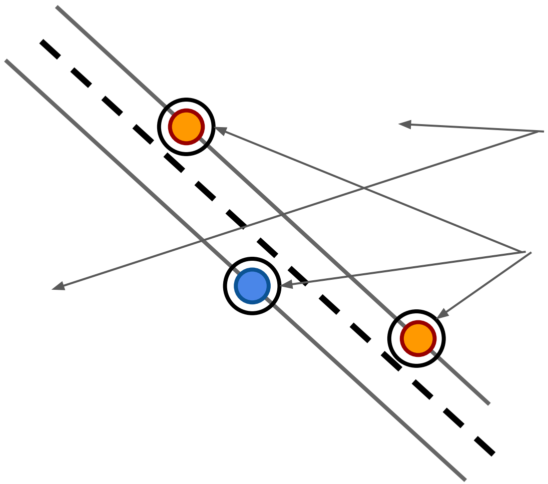
$$\mathbf{w}^\mathbf{T}\mathbf{x} + b \geqslant +1$$
$$\mathbf{w}^\mathbf{T}\mathbf{x} + b \leqslant -1$$

$$\Rightarrow$$

$$y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geqslant +1$$

$$y_i = +1 \text{ for positives}$$
$$y_i = -1 \text{ for negatives}$$

$$\Downarrow$$



$$y_i(\mathbf{w}^\mathbf{T}\mathbf{x} + b) - 1 \geqslant 0$$

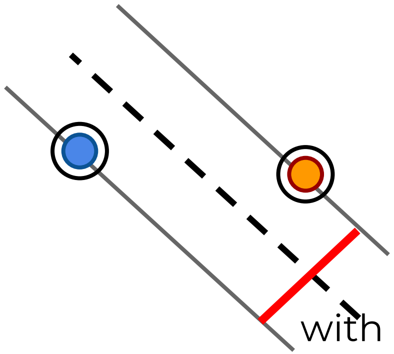
(for any point)

$$y_i(\mathbf{w}^\mathbf{T}\mathbf{x} + b) - 1 = 0$$

(for the support vectors)

Support Vector Machines

- **Width of the margin**



Using the previous equation $y_i(\mathbf{w}^\mathbf{T}\mathbf{x} + b) - 1 = 0$ and two arbitrary support vectors in both sides of the margin, we can derive that the width of the margin is $\frac{2}{||\mathbf{w}||}$

We want to **maximize** this width!

Which is the same as: we want to **minimize** $\frac{1}{2}||\mathbf{w}||^2$

(for mathematical convenience)

Support Vector Machines

- **How to find extrema of a function with constraints?**

  Lagrange Multipliers

Constraints (all the points)

$$L = \frac{1}{2}||\mathbf{w}||^2 - \sum_i \alpha_i [y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) - 1]$$

We want to maximize the width of the margin

This is a Lagrange multiplier

Constrained to the aforementioned condition

Support Vector Machines

- **How to find extrema of a function with constraints?**

  Solving the Lagrange Multipliers

$$L = \frac{1}{2}||\mathbf{w}||^2 - \sum_i \alpha_i [y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) - 1]$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x_i} = 0 \Rightarrow \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x_i}$$

$$\frac{\partial L}{\partial b} = -\sum_i \mathbf{x_i} y_i = 0 \Rightarrow \sum_i \mathbf{x_i} y_i = 0$$

Support Vector Machines

- **How to find extrema of a function with constraints?**

Putting all together and deriving we reach to this equation to optimize:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \underbrace{\mathbf{x_i} \mathbf{x_j}}$$

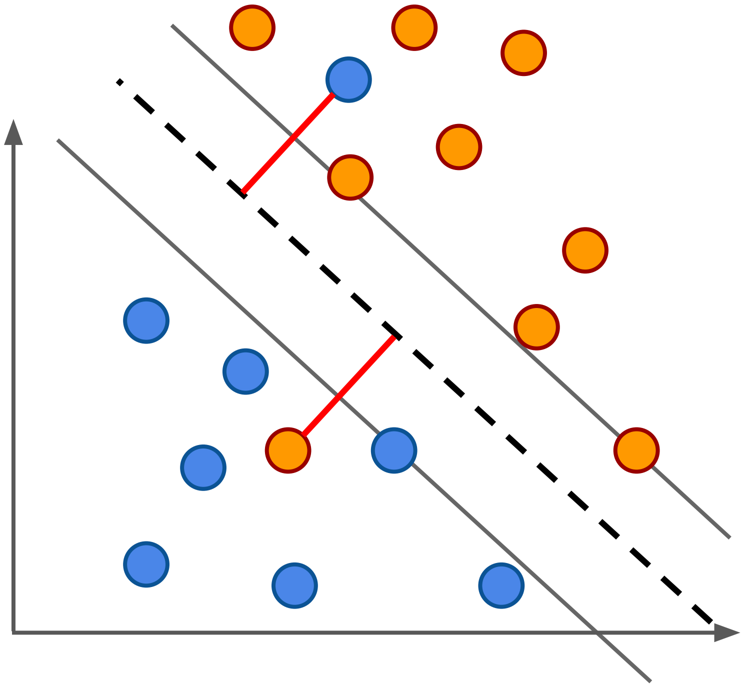
Interestingly, it depends only on **dot products of samples**

More interestingly, the **decision rule** to classify the sample $\mathbf{x_u}$ also depends only on the dot product of samples:

$$\sum_i \alpha_i y_i \mathbf{x_i} \mathbf{x_u} + b \geqslant 0$$

Support Vector Machines

- **Now what if not fully linearly separable?**



This was the original decision rule

$$y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geqslant 1$$

Add a slack variable

$$y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geqslant 1 - \xi_i$$

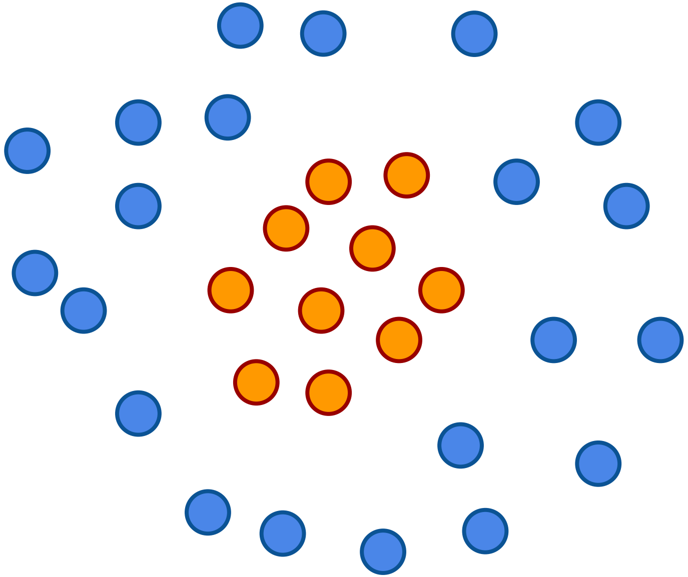
And incorporate it.in the optimization

$$C \sum_i \xi_i$$

(C sets how strict are we to outliers)

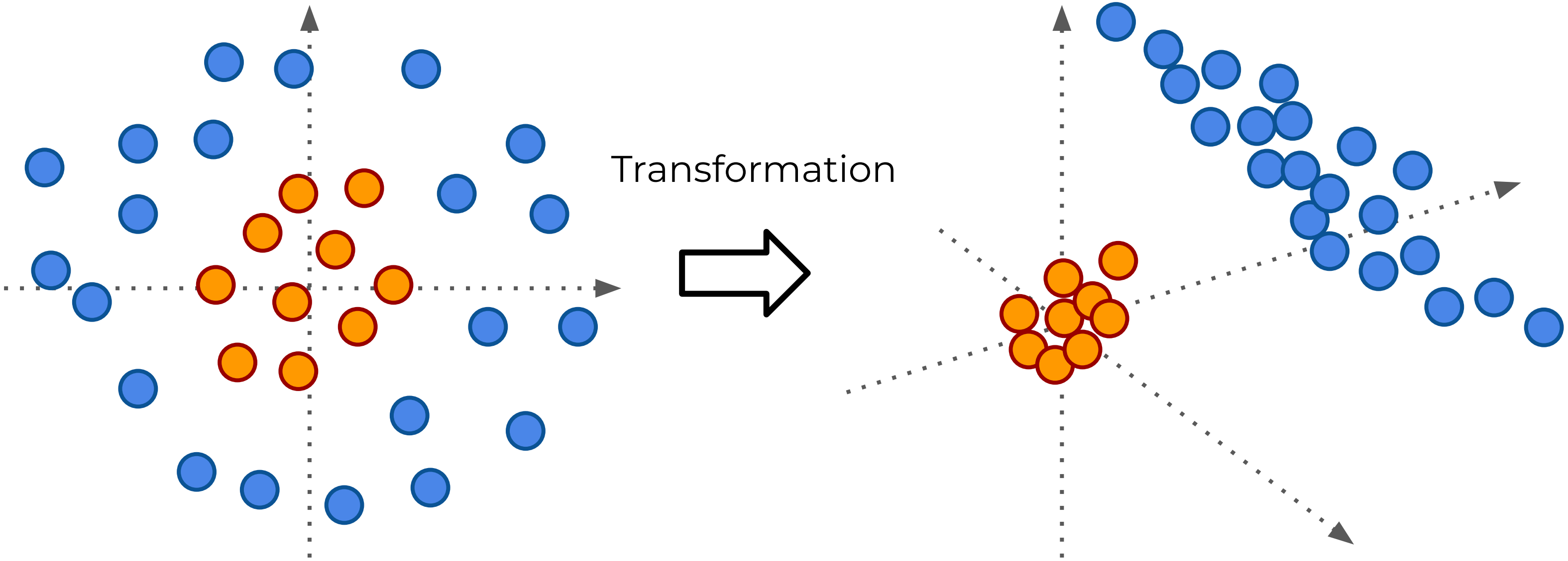
Support Vector Machines

- **What if non-linearly separable at all?**



Support Vector Machines

- **Project the data into a higher dimensional space to make it linearly separable**



$$\mathbf{x} = \{x_1, x_2\}$$

$$\Phi(\mathbf{x}) = \{x_1^2, x_2^2, \sqrt{2}x_1x_2\}$$

Support Vector Machines

- **The kernel trick**

This was the decision rule

$$\sum_i \alpha_i y_i \mathbf{x_i} \mathbf{x_u} + b \geqslant 0$$

Now if we use transformations, it becomes:

$$\sum_i \alpha_i y_i \Phi(\mathbf{x_i}) \Phi(\mathbf{x_u}) + b \geqslant 0$$

$$K(\mathbf{x_i}, \mathbf{x_u}) = \Phi(\mathbf{x_i}) \Phi(\mathbf{x_u})$$

We do not even need to know Φ, but the results of the dot product of the transformations

(This applies also to the optimization equation)

Support Vector Machines

- **The kernel trick**

This was the decision rule

$$\sum_i \alpha_i y_i \mathbf{x_i} \mathbf{x_u} + b \geqslant 0$$

Now if we use transformations, it becomes:

$$\sum_i \alpha_i y_i \Phi(\mathbf{x_i}) \Phi(\mathbf{x_u}) + b \geqslant 0$$

$$K(\mathbf{x_i}, \mathbf{x_u}) = \Phi(\mathbf{x_i}) \Phi(\mathbf{x_u})$$

Less operations
Less spatial complexity

We do not even need to know Φ, but the results of the dot product of the transformations

(This applies also to the optimization equation)

Support Vector Machines

- **The kernel trick**

Instead of manually defining transforms , we just play with dot products:

w/o kernel trick

Define, $\Phi(\mathbf{x}) \rightarrow 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$.

$\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$

$= \langle \{1, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}, x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2}\}, \{1, \sqrt{2}x_{j1}, \sqrt{2}x_{j2}, x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2}\} \rangle \quad (6.1)$

$= 1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} \quad (6.2)$

w/ kernel trick

$(1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^2$

$= (1 + x_{i1}x_{j1} + x_{i2}x_{j2})^2 \quad (7.1)$

$= 1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} \quad (7.2)$

(Example from https://www.quora.com/What-is-the-kernel-trick)

Support Vector Machines

- **The kernel trick**

  Some examples of kernels are:

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^p; \text{ polynomial kernel.}$$

$$K(x_i, x_j) = e^{\frac{-1}{2\sigma^2}(x_i - x_j)^2}; \text{ Gaussian kernel; Special case of Radial Basis Function.}$$

$$K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}; \text{ RBF Kernel}$$

$$K(x_i, x_j) = \tanh(\eta\, x_i \cdot x_j + \nu); \text{ Sigmoid Kernel; Activation function for NN.}$$