Kernels, dual form, etc.

1. Kernelized Perceptron:

- First, recall the exact sol° for Lin Reg:

$$w = (\underbrace{X^T X}_{C,\ \text{cov. matrix of the data}})^{-1} X\ Y$$

$$= C^{-1} X.Y$$

$$= \sum_n x_n y_n \cdot (C^{-1})$$

- This is quite different from the perceptron solution: Starting from $\vec{w} = \vec{0}$, $b = 0$, we add $+\eta\, \vec{x}_n y_n$ to $\vec{w}$ each time an example $(\vec{x}_n, y_n)$ is wrongly classified by $\vec{w}$ (think eg of the Online version).
So, assuming we do converge (as is guaranteed by the model in the case of linearly separable data), the solut° $\vec{w}^*$ is built as a sum like:

$$\vec{w} = \Big(\sum_n \underbrace{\alpha_n}_{\downarrow} \vec{x}_n y_n\Big) \eta$$

$\begin{cases} 0 \text{ if it was always well classif.,} \\ 1 \text{ if it was incorrectly classified only exactly once} \\ 2 \end{cases}$

$\hookrightarrow \alpha_n$ is a counter (≥ 0) of the number of times example n was misclassified

We drop η or include it in α_n, so that we have $\vec{w} = \sum_n \alpha_n \vec{x}_n y_n$

In this view, $\vec{w}$ is a linear combinat° of the training examples $\vec{x}_n$, with weights $\alpha_n y_n$ ($\alpha_n > 0$, $y_n = \pm 1$).

This may be called the dual form.

In the online perceptron, in a sense, we are updating the α_n's (they start from $\alpha_n = 0, \forall n$).

• (This is very ≠ from $w = \sum_n C^{-1} \vec{x}_n y_n$ in linear reg, where in effect, $\alpha_n = C^{-1}, \forall n$, (constant α_n).)

• At <u>prediction time</u>, we get:

$$y^{pred}(\vec{x}^{test}) = \text{sign}(\vec{w}\,\vec{x}_{test})$$

$$= \text{sign}\Big(\sum_n \alpha_n y_n \underline{\vec{x}_n \cdot \vec{x}_{test}}\Big)$$

If we used feature maps, $\phi\, x_n \to \phi(x_n)$, we would have: $y_{test} = \text{sign}\Big(\sum_n \alpha_n y_n \underline{\phi(\vec{x}_n)\,\phi(\vec{x}_{test})}\Big)$

• <u>Remark</u>: $\vec{x}_n\,\vec{x}_{test}$ is a measure of the similarity between $\vec{x}_n$ and $\vec{x}_{test}$. If they're very ≠, it's ≃ 0, and α_n does not matter for y_{test}. If they are very similar, it is large, and α_n matters.
(Note: if data is standardized, then it cannot grow too large).

• <u>Definition</u>: We call <u>Kernel method</u> the fact of replacing $\vec{x}\cdot\vec{x}'$ (or $\phi(\vec{x})\cdot\phi(\vec{x}')$) with an other function $K(\vec{x}, \vec{x}')$, which is called a Kernel.

Several remarks:

1) Kernels are more general than feature maps:

- all feature maps are Kernels:

  $K(x, x') = \phi(x)\,\phi(x')$ is a Kernel for any feature map ϕ.

- not all Kernels can be re-written as feature maps (see eg the RBF Kernel $\rightarrow$ it's like a $D = \infty$ feature map)

- Not all funct° of 2 variables are valid Kernels

$\downarrow$

We must respect the Mercer condit° :

Mercer condition = K must be a semi-positive definite operator

$= \forall f \in L^2(\mathbb{R}), \forall g \in L^2(\mathbb{R}), \quad \int_{x,y} f(x)\, K(x,y)\, g(y) \, dx\, dy \geqslant 0$

In a discrete setting, this would be like $\forall f \in \mathbb{R}^d, \forall g \in \mathbb{R}^d$, we must have : $\quad f^T K g = \sum_i f_i K_{ij} g_j \geqslant 0$.

- Remark how K builds a new geometry in the space of features: the similarity between two points, instead of being measured by $\vec{x} \cdot \vec{x}'$ of $1/\|x - x'\|_2^2$, is measured by $K(x, x')$.