$$\underbrace{k(\overset{\text{inputs}}{x}, z)}_{\text{kernel function}}$$ measures similarity between $x$ and $z$

high similarity $\Rightarrow$ large $k(x,z)$

Logistic regression is already computing prediction given $x$ based on

$$\sum_{i=1}^{n} a_i k(x, x^{(i)}) \begin{cases} \text{if } > 0, \text{ predict } +1 \\ \text{if } < 0, \text{ predict } -1 \end{cases}$$

if we define $k(x, x^{(i)}) = w^T z$

for each training example, compute its similarity to $x$ and multiply by a learned weight $a_i$

original logistic regression

1) Training: $w^{(t)} \leftarrow w^{(t-1)} + \eta \sum_{i=1}^{n} \underbrace{\sigma(-y^{(i)} w^{(t-1)T} x^{(i)}) y^{(i)}}_{\text{scalar}} \cdot x^{(i)}$

2) Testing: compute $w^T x$

Kernel logistic regression (equivalent mathematically)

Define $a \in \mathbb{R}^n$, $w = \sum_{i=1}^{n} a_i x^{(i)}$

1) Training: $a_i^{(t)} \leftarrow a_i^{(t-1)} + \eta \cdot \sigma(-y^{(i)} \underbrace{w^{(t-1)T} x^{(i)}}_{= \sum_{j=1}^{n} a_j x^{(j)T} x^{(i)}}) \cdot y^{(i)}$

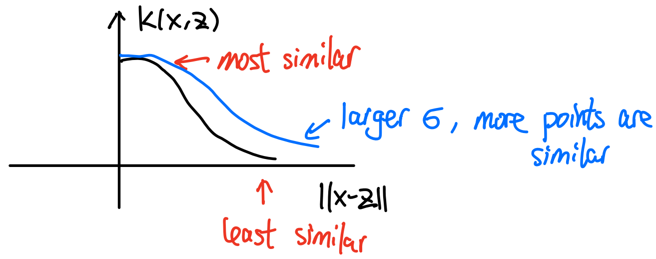
$$= a_i^{(t-1)} + \eta \cdot \sigma(-y^{(i)} \sum_{j=1}^{n} a_j \underbrace{k(x^{(j)}, x^{(i)})}_{\text{only place we use } x\text{'s}}) \cdot y^{(i)}$$

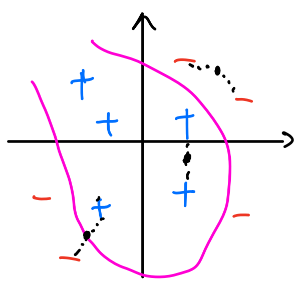
2) Testing: compute $\sum_{j=1}^{n} a_j k(x^{(j)}, x)$

We can run the algorithm with any choice of the kernal function.

## Radial Basis Function (RBF)

$$k(x, z) = \exp\left(-\frac{\|x-z\|^2}{\underbrace{2\sigma^2}_{\text{hyperparam}}}\right)$$



In practice, RBF is a very popular way to learn non-linear decision boundary.

## Kernals vs Features

| y | $x_1$ | $x_2$ |
|---|---|---|
| +1 | 2 | 3 |
| -1 | 0 | 1 |

transform each row →

| y | $x_1$ | $x_2$ | 1 | $x_1^2$ | $x_2^2$ | $x_1x_2$ |
|---|---|---|---|---|---|---|
| +1 | 2 | 3 | 1 | 4 | 9 | 6 |
| -1 | 0 | 1 | 1 | 0 | 1 | 0 |

call this transformation $\phi: \mathbb{R}^2 \to \mathbb{R}^6$

We can run normal logistic regression with $\phi$,
but it would take 3x long.

"kernal trick"

Use a kernalized algorithm so

$$k(x, z) = \phi(x)^T \phi(z)$$

In many cases can compute this directly
(without creating $\phi(x)$, $\phi(z)$)

Polynomial kernel for degree 2 (Quadratic kernal)

$k(x,z) = (x^T z + 1)^2 = \phi(x)^T \phi(z)$

when $\phi(x) = \begin{bmatrix} 1 \\ \sqrt{2} x_1 \\ \sqrt{2} x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{bmatrix}$

You can compute $\phi(x)^T \phi(z)$ without running $\phi$

In general, for any original dimension of x's for any degree p.

$$k(x,z) = (x^T z + 1)^p = \phi(x)^T \phi(z)$$

for some $\phi$ that has all monomials of degree $\leq p$

What about RBF?

Fact: $\exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right) = \phi(x)^T \phi(z)$

for some $\phi(x)$ that is infinite dimension

Kernal logistic regression + RBF (doable)

$\Updownarrow$

logistic regression + d-dimensional features (not doable)

Runtime: polynomial kernal degree p

Original : T iterations, each $O(n \cdot \underbrace{d^p}_{\text{size of } \phi(x)})$

kernal : T iterations, each $O(n^2 \cdot \underbrace{d}_{\text{computing } k(x,z)})$

kernal pay $O(n^2)$, but enables using more features at no additional costs