2/9/2023: Kernels (Part 2)

inputs to predictor

$\underbrace{K(x, z)}_{\text{kernel function}}$ measures similarity between x & z

- similar x & z should have large $K(x,z)$

Logistic Regression is already computing prediction given input x based on

$$\boxed{\sum_{i=1}^{n} a_i K(x, x^{(i)})} \quad \begin{array}{l} \to \text{If } > 0 \text{ predict } +1 \\ \to \text{If } < 0 \text{ predict } -1 \end{array}$$

If we define $K(x,z) = x^T z$

For each training example, compute its similarity to x & multiply by a learned weight a_i

Original Logistic Regression

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

For example (n=3): $+ \underline{0.7}\, x^{(1)} - \underline{0.3}\, x^{(2)} + \underline{0.2}\, x^{(3)}$

2) Testing: we compute $w^T x$

Kernel Logistic Regression (Equivalent mathematically)

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

1) Training: $a_i^{(t)} \leftarrow a_i^{(t-1)} + \eta \cdot \sigma(-y^{(i)} \boxed{w^{(t-1)T} x^{(i)}}) \cdot y^{(i)}$

For example:

$a_1^{(t)} \leftarrow a_1^{(t-1)} + \underline{0.7}$

$a_2^{(t)} \leftarrow a_2^{(t-1)} - \underline{0.3}$

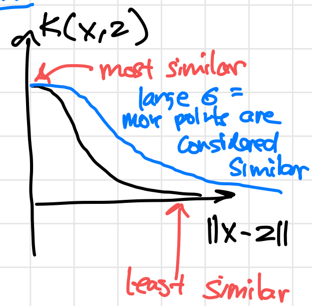
$a_3^{(t)} \leftarrow a_3^{(t-1)} + \underline{0.2}$

$= \sum_{j=1}^{n} a_j x^{(j)T} x^{(i)}$

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

2) Similarly, testing: compute $\sum_{j=1}^{n} a_j \underbrace{K(x^{(j)}, x)}_{\text{only place } x \text{ is used}}$

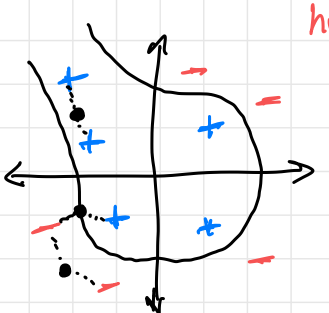
Big payoff: we can run kernelized algorithm with <u>any choice of kernel function</u>



Radial Basis Function (RBF)

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\boxed{\sigma^2}}\right)$$

hyperparameter



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

Kernels & Features

y	x_1	x_2
+1	2	3
-1	0	1

transform each row →

y	$\sqrt{2}x_1$	$\sqrt{2}x_2$	1	x_1^2	x_2^2	$\sqrt{2}x_1x_2$
+1	$2\sqrt{2}$	$3\sqrt{2}$	1	4	9	$6\sqrt{2}$
-1	0	$1\sqrt{2}$	1	0	1	0

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

We can run normal Logistic Regression with ϕ, but it would take 3x long

Idea: "Kernel trick"

Use a kernelized algorithm, so

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

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

Polynomial Kernel for degree 2 (Quadratic Kernel)

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

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" ϕ

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

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

for some ϕ that has all monomials of degree $\le p$

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 dimensional.

Kernel Logistic regression + RBF (doable)

equivalent to

Logistic regression + ∞-dimensional features (not doable)

Runtime Considerations: Polynomial kernel, degree p, original dimension is d

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

Kernel: T iterations, each $O(n^2 \cdot d)$ computing $K(x,z)$

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