Kernel Methods

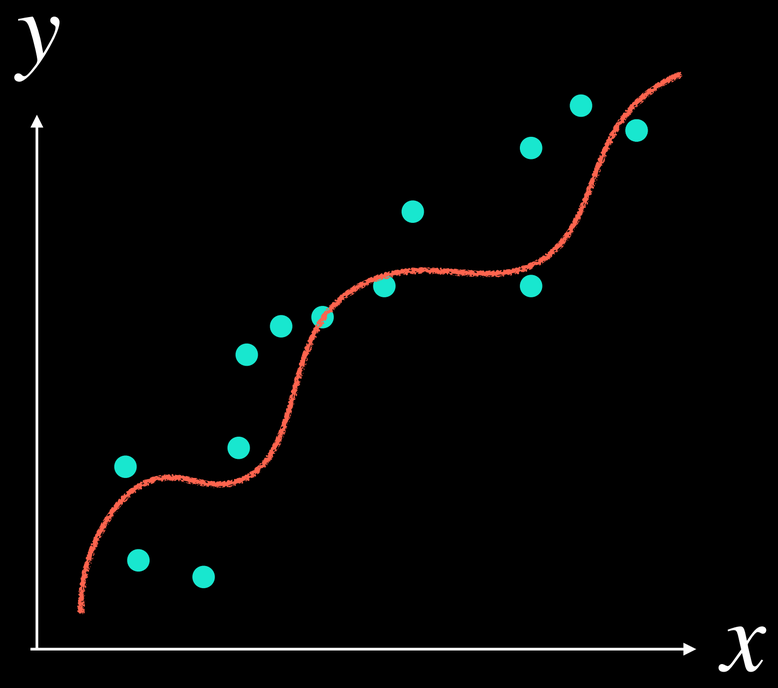
Feature Engineering

h is does not have to be linear in x

Input	Output
x	y
$x^{(1)}$	$y^{(1)}$
$x^{(2)}$	$y^{(2)}$
$x^{(3)}$	$y^{(3)}$
$x^{(4)}$	$y^{(4)}$

Example: construct a polynomial model

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \dots$$



$$h_\theta(x) = \underbrace{[\theta_0, \theta_1, \theta_2, \theta_3, \dots]}_{\theta} \cdot \underbrace{[1, x, x^2, x^3, \dots]}_{\phi(x)}$$

Feature map

$$h_\theta(x) = \theta^\top \phi(x)$$

x are sometimes called input **attributes**

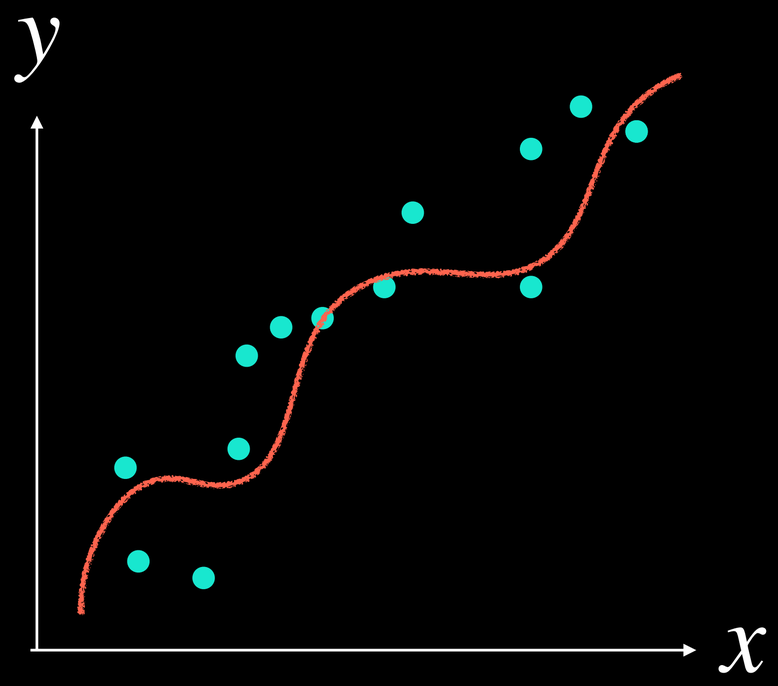
Least mean squares

Input	Output
x	y
$x^{(1)}$	$y^{(1)}$
$x^{(2)}$	$y^{(2)}$
$x^{(3)}$	$y^{(3)}$
$x^{(4)}$	$y^{(4)}$

Gradient Descent Update

$$\theta := \theta + \alpha \left(y^{(i)} - h_\theta \left(x^{(i)} \right) \right) x^{(i)}$$

$$\theta := \theta + \alpha \left(y^{(i)} - \theta^\top \phi \left(x^{(i)} \right) \right) \phi \left(x^{(i)} \right)$$



Least mean squares

$\phi(x) \in \mathbb{R}^p$

What happens if p is large?

Polynomial of degree 5
with a $d = 100$ attributes: $x \in \mathbb{R}^{100}$

Requires storing 10^{10} dimensional vector for θ

Can we simplify the problem to store less values?

$$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_1^2 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_2 x_1 \\ \vdots \\ x_1^3 \\ x_1^2 x_2 \\ \vdots \end{bmatrix}$$

Kernel Trick

θ can be represented as a linear combination of $\phi\left(x^{(1)}\right), \phi\left(x^{(2)}\right), \ldots, \phi\left(x^{(n)}\right)$

$$\theta = \sum_{i=1}^{n} \beta_i \phi\left(x^{(i)}\right)$$

$\beta_1, \ldots, \beta_n \in \mathbb{R}$

Kernel Trick

$$\theta = \sum_{i=1}^{n} \beta_i \phi\left(x^{(i)}\right)$$

Replacing in the update rule:

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)})$$

$$:= \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)})$$

$$:= \sum_{i=1}^{n} \left(\beta_i + \alpha \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)\right) \phi(x^{(i)})$$

new β_i

Kernel Trick

$$\theta = \sum_{i=1}^{n} \beta_i \phi\left(x^{(i)}\right)$$

Replacing in the update rule:

$$\theta := \sum_{i=1}^{n} \underbrace{\left(\beta_i + \alpha\left(y^{(i)} - \theta^T \phi(x^{(i)})\right)\right)}_{\text{new } \beta_i} \phi(x^{(i)})$$

$$\beta_i := \beta_i + \alpha\left(y^{(i)} - \theta^T \phi(x^{(i)})\right)$$

$$\beta_i := \beta_i + \alpha\left(y^{(i)} - \sum_{j=1}^{n} \beta_j \, \phi(x^{(j)})^T \phi(x^{(i)})\right)$$

For all $i = 1 \ldots n$

Kernel Trick

$$\theta = \sum_{i=1}^{n} \beta_i \phi\left(x^{(i)}\right)$$

New update rule, in β instead of θ:

$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j \, \phi(x^{(j)})^T \phi(x^{(i)}) \right)$$

Dot product

For all $i = 1 \ldots n$

Don't we still need to compute a dot product of 2 $\mathbb{R}^p$ (p is large) pairs $\forall (i, j)$?

- We can precompute $\phi(x^{(j)}) \cdot \phi(x^{(i)})$ for all pairs before running SGD
- For some feature maps $\phi()$, computing $\phi(x^{(j)}) \cdot \phi(x^{(i)})$ can be efficient and doesn't require computing $\phi(x^{(i)})$

Kernel Trick

Computing $\phi(x^{(j)}) \cdot \phi(x^{(i)})$ doesn't require computing $\phi(x^{(i)})$

$$\langle \phi(x), \phi(z) \rangle = 1 + \sum_{i=1}^{d} x_i z_i + \sum_{i,j \in \{1,\ldots,d\}} x_i x_j z_i z_j + \sum_{i,j,k \in \{1,\ldots,d\}} x_i x_j x_k z_i z_j z_k$$

$$= 1 + \sum_{i=1}^{d} x_i z_i + \left(\sum_{i=1}^{d} x_i z_i \right)^2 + \left(\sum_{i=1}^{d} x_i z_i \right)^3$$

$$= 1 + \langle x, z \rangle + \langle x, z \rangle^2 + \langle x, z \rangle^3$$

No need to compute $\phi(x)$

$$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_1^2 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_2 x_1 \\ \vdots \\ x_1^3 \\ x_1^2 x_2 \\ \vdots \end{bmatrix}$$

Kernel Trick

Computing $\phi(x^{(j)}) \cdot \phi(x^{(i)})$ doesn't require computing $\phi(x^{(i)})$

$$\langle \phi(x), \phi(z) \rangle = 1 + \sum_{i=1}^{d} x_i z_i + \sum_{i,j \in \{1,\ldots,d\}} x_i x_j z_i z_j + \sum_{i,j,k \in \{1,\ldots,d\}} x_i x_j x_k z_i z_j z_k$$

$$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_1^2 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_2 x_1 \\ \vdots \\ x_1^3 \\ x_1^2 x_2 \\ \vdots \end{bmatrix}$$

Kernel Trick

Definition of a **Kernel** corresponding to a feature map ϕ is:

$$K(x,z) = \phi(x) \cdot \phi(z) \qquad = \langle \phi(x), \phi(z) \rangle$$

Algorithm

Compute: $K\left(x^{(i)}, x^{(j)}\right) = \langle \phi\left(x^{(i)}\right) \cdot \phi\left(x^{(j)}\right) \rangle$ for all $i, j \in \{1,...,n\}$

Loop: $\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j K\left(x^{(i)}, x^{(j)}\right) \right)$ for all $i \in \{1,...,n\}$

$$\beta := \beta + \alpha \left(\vec{y} - K\beta \right)$$

Hypothesis with Kernels

How do we find compute the predictor?

$$h_\theta(x) = \theta^T \phi(x)$$

$$= \sum_i^n \beta_i \, \phi\left(x^{(i)}\right)^\top \phi(x)$$

$$= \sum_i^n \beta_i \, K\left(x^{(i)}, x\right)$$

All we need to know about ϕ is in K

Other Kernels?

What kind of Kernel functions $K(\cdot, \cdot)$ correspond to some feature map ϕ?

Instead of picking ϕ, can we pick a kernel function K that satisfies the property that $K(x, z) = \phi(x) \cdot \phi(z)$?

Example

$$K(x, z) = (x^T z)^2$$

Other Kernels?

What kind of Kernel functions $K(\cdot, \cdot)$ correspond to some feature map ϕ?

Instead of picking ϕ, can we pick a kernel function K that satisfies the property that $K(x, z) = \phi(x) \cdot \phi(z)$?

$$K(x, z) = (x^T z)^2$$

Kernels as similarity metrics

What kind of Kernel functions $K(\cdot, \cdot)$ correspond to some feature map ϕ?

$K = \phi(x) \cdot \phi(z)$ can be considered a distance metric between $\phi(x)$ and $\phi(z)$

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

As long as K is positive semidefinite, it's a valid kernel

Example

[0, 0, 1, 0, 0, 0, 0, 0, 0]