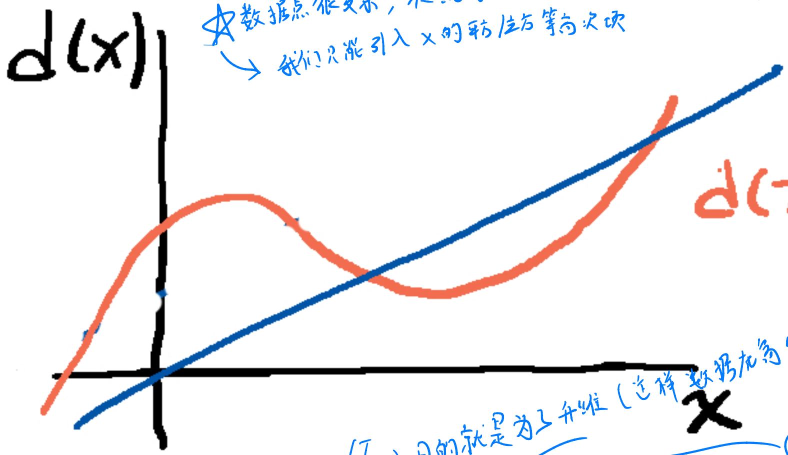
Kernel Regression

Objectives

- Why use higher-dimensional feature spaces
- Reformulate regression in terms of kernels
- Popular kernels
- Cautions and considerations

Higher dimensional feature spaces extend regression

★数据点很复杂，传统的线性回归没法完美拟合数据
→ 我们只能引入 x 的高次方等高次项



$$d(x) = w_1 x$$

d(x) 是决策函数而不是 cost function（拿来一个 x 算出来我们要的预测值）

$$d(x) = w_3 x^3 + w_2 x^2 + w_1 x + w_0$$

Let $\underline{x} = [x_1 \; x_2 \; \cdots \; x_M]^T \in \mathbb{R}^M$ (M features)

Consider $d(\underline{x}) = \underline{\phi}^T(\underline{x})\,\underline{w}$, $\phi(\underline{x}) \in \mathbb{R}^P$, $P > M$

$\phi^T(x)$ 目的就是为了升维（这样数据在高维空间就会好分）

Example: $\underline{x} = [x_1 \; x_2]^T$, $\underline{\phi}^T(\underline{x}) = [x_1^2 \;\; x_2^2 \;\; \sqrt{2}\,x_1 x_2 \;\; x_1 \;\; x_2 \;\; 1]$

Finding $\underline{w}$: "training" data $\underline{x}^i, d^i, \; i = 1, 2, \cdots N$

$$\min_{\underline{w}} \sum_{i=1}^{N} \left(d^i - \underline{\phi}^T(\underline{x}^i)\,\underline{w}\right)^2 + \lambda \|\underline{w}\|_2^2 \quad \text{(Ridge)}$$

$\underline{d} = [d^1 \; d^2 \; \cdots \; d^N]^T$ (N samples)

$\underline{\Phi} = [\phi(\underline{x}^1) \;\; \phi(\underline{x}^2) \;\; \cdots \;\; \phi(\underline{x}^N)]^T$ (N samples) $(N \times P)$

$$\min_{\underline{w}} \|\underline{d} - \underline{\Phi}\,\underline{w}\|_2^2 + \lambda \|\underline{w}\|_2^2$$

$$\Rightarrow \underline{w} = (\underline{\Phi}^T \underline{\Phi} + \lambda \underline{I})^{-1} \underline{\Phi}^T \underline{d}$$

Regression is a weighted sum of "kernels"

$$d(\underline{x}) = \underline{\phi}^T(\underline{x})\,\underline{w} = \underline{\phi}^T(\underline{x})\left(\underline{\Phi}^T\underline{\Phi} + \lambda \underline{I}\right)^{-1}\underline{\Phi}^T\underline{d}$$

$\underline{\Phi}^T$ is $P \times N$, $\underline{\Phi}$ is $N \times P$ → $\underline{\Phi}^T\underline{\Phi}$ is $P \times P$

Matrix identity: $\left(\underline{\Phi}^T\underline{\Phi} + \lambda \underline{I}\right)^{-1}\underline{\Phi}^T = \underline{\Phi}^T\left(\underline{\Phi}\,\underline{\Phi}^T + \lambda \underline{I}\right)^{-1}$ (activity)

Thus $d(\underline{x}) = \underline{\phi}^T(\underline{x})\,\underline{\Phi}^T\left(\underline{\Phi}\,\underline{\Phi}^T + \lambda \underline{I}\right)^{-1}\underline{d}$

$N \times N$ → 样本数 × 样本数（如果小的话运算量就非常小了）

Note: $\left[\underline{\Phi}\,\underline{\Phi}^T\right]_{i,j} = \underline{\phi}^T(\underline{x}^i)\,\underline{\phi}(\underline{x}^j)$

$\underline{\Phi}$ 和 $\underline{\Phi}^T$ 的积的 ij 元素 = ith and jth training sample's dot product

$$\left[\underline{\phi}^T(\underline{x})\,\underline{\Phi}^T\right]_j = \underline{\phi}^T(\underline{x})\,\underline{\phi}(\underline{x}^j)$$

新数据和 $\underline{\Phi}$ 的乘积 = 新数据和 jth training sample $\underline{x}^j$'s dot product

Define "Kernel" 核函数（向量点积代表相似度）

$$K(\underline{u},\underline{v}) = \underline{\phi}^T(\underline{u})\,\underline{\phi}(\underline{v})$$

(compute similarity)

不关心复杂的特征具体是多少，只关心两个 feature pair $\underline{u}$ 和 $\underline{v}$ 在高维空间内的乘积

Let

$$\underline{\alpha} = \left[\alpha_1 \cdots \alpha_N\right]^T \to N \times 1$$

$$= \left(\underline{\Phi}\,\underline{\Phi}^T + \lambda \underline{I}\right)^{-1}\underline{d} \to N \times 1$$

$N \times N$, $N \times 1$

$d(x) = \phi^T(x)\,\underline{\Phi}^T\,\alpha$

$$d(\underline{x}) = \sum_{i=1}^{N} \alpha_i\,\underline{\phi}^T(\underline{x})\,\underline{\phi}(\underline{x}^i) = \sum_{i=1}^{N} \alpha_i K(\underline{x},\underline{x}^i)$$

新数据和每个数据的相似度

Kernel methods find $d(\underline{x})$ without computing $\phi(\underline{x})$

$$d(\underline{x}) = \sum_{i=1}^{N} \alpha_i K(\underline{x}, \underline{x}^i)$$

$$\underline{\alpha} = (\underline{\Phi}\,\underline{\Phi}^T + \lambda \underline{I})^{-1} \underline{d} \quad \rightarrow \quad \underline{K} = \underline{\Phi}\,\underline{\Phi}^T$$

$$= (\underline{K} + \lambda \underline{I})^{-1} \underline{d}$$

$\underline{K}$ ↓ Gram matrix

用 K 代表 $\Phi\Phi^T$

$$[\underline{K}]_{ij} = \underline{\phi}^T(\underline{x}^i)\, \underline{\phi}(\underline{x}^j) = K(\underline{x}^i, \underline{x}^j)$$

☆ 数学定义上的，但是我们实际计算用的是下面的 kernel trick → 在原始低维空间做点积（绕过 $\phi(x)$）

$\underline{K}$ can be computed efficiently!

Ex: Monomials of degree q $\quad \underline{\phi}(\underline{x}) \rightarrow \underbrace{x_1^q,\ x_1^{q-1} x_2,\ \ldots\ x_3^{q-5} x_6^2 x_8^3 \ldots}$

$$K(\underline{u}, \underline{v}) = \underbrace{\underline{\phi}^T(\underline{u})\, \underline{\phi}(\underline{v})}_{O(P)} = (\underbrace{\underline{u}^T \underline{v}}_{O(M)})^q \quad \text{(activity)}$$

$$P = \frac{(q + M - 1)!}{q!\,(M-1)!} \text{ terms}$$

P ↑ 扩展后的特征总数

q ← 要扩展的次数

M → 原始特征数

Suppose $M = 10,\ q = 5 \rightarrow P \sim 2000$ $\quad$ computing $O(P)$ vs $O(M)$

$M = 100,\ q = 5 \rightarrow P \sim 10^8$ $\quad$ memory $O(NP)$ vs $O(N^2)$

Popular kernels depend on similarity of $\underline{u}, \underline{v}$

$$\underline{u}^T \underline{v} = \|\underline{u}\|_2 \|\underline{v}\|_2 \cos\theta$$

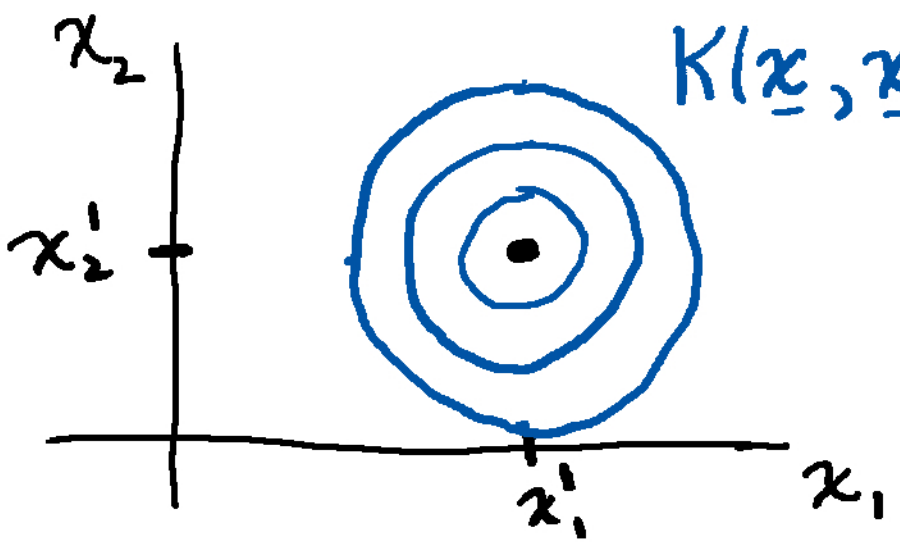
☆定义好的，直接用

Monomials of degree q: $K(\underline{u}, \underline{v}) = (\underline{u}^T \underline{v})^q$

Polynomials up to degree q: $K(\underline{u}, \underline{v}) = (\underline{u}^T \underline{v} + 1)^q$

Gaussian/radial kernel: $K(\underline{u}, \underline{v}) = \exp\left\{ - \frac{\|\underline{u} - \underline{v}\|_2^2}{2\sigma^2} \right\}$

sigma



- No explicit $\underline{\phi}(\underline{x})$
- All polynomial orders
- smoothness controlled by σ

Kernel regression considerations

$$d(\underline{x}) = \underline{\phi}^T(\underline{x})\,\underline{w} \quad \text{vs} \quad d(\underline{x}) = \sum_{i=1}^{N} \alpha_i\, k(\underline{x}, \underline{x}^i)$$

原始视角　　核方法视角

- Store and compute $\underline{\alpha}$ (Nx1) vs $\underline{w}$ (Px1)
- Binary classification　$\text{sign}\{d(\underline{x})\}$

可以把回归变成分类

- Avoid "overfitting" with high-D feature spaces

(cross-validation)