Lecture 11 :

Kernel Methods

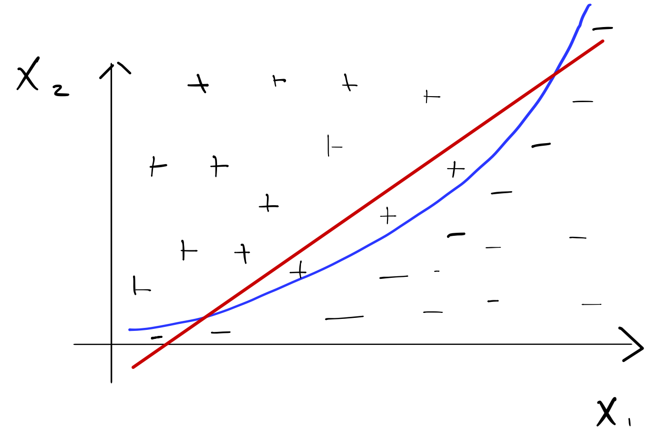
Kernel Regression

so far: $\hat{y} = w_0 + w_1 x_1 + w_2 x_2 = w^T \underline{x}$ $(p=3)$

new: $\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2$ $(d=6)$

Let $\phi(\underline{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix} \Rightarrow \hat{y} = \underline{w}^T \phi(\underline{x})$

$\phi(\underline{x}) \in \mathbb{R}^6$



Kernel methods formalize and generalize this idea

original feature vector $\underline{x} \in \mathbb{R}^p \xrightarrow{\text{map to}} \phi(\underline{x}) \in \mathbb{R}^d$ "high-dimensional feature space"

typically $d > p$

for degree 5 polynomials:

$$\phi(\underline{x}) = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_p \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_1 x_p \\ x_2 x_3 \\ \vdots \\ x_p^2 \\ x_1 x_2 x_3 \\ x_1 x_2 x_4 \\ \vdots \\ x_1 x_2 x_3 x_4 \\ x_1 x_2 x_3 x_5 \\ \vdots \\ x_1 x_2 x_3 x_4 x_5 \\ x_1 x_2 x_3 x_4 x_6 \\ \vdots \end{bmatrix}$$

$d \approx p^5$

goal: find weights s.t. $\hat{y} = \hat{\underline{w}}^T \phi(\underline{x})$ gives good predictions

Kernel methods allow us to do this <u>without computing $\phi(x_i)$ explicitly</u> for all training samples $\underline{x}_i$

this is especially helpful because in many cases $d \to \infty$

How do we do this?

Recall Ridge Regression

$$\min_{\underline{w}} \underbrace{\|\underline{y} - X\underline{w}\|_2^2 + \lambda \|\underline{w}\|_2^2}_{f(\underline{w})} . \qquad \text{Let } X = U\Sigma V^T .$$

Compute gradient and set to zero:

$$\Rightarrow \hat{\underline{w}} = (X^TX + \lambda I)^{-1} X^T \underline{y}$$

$$= V \underbrace{(\Sigma^T\Sigma + \lambda I)^{-1}\Sigma^T}_{} U^T \underline{y}$$

$$= \left[\begin{array}{ccc|c} \frac{\sigma_1}{\sigma_1^2+\lambda} & & & \\ & \ddots & & 0 \\ & & \frac{\sigma_p}{\sigma_p^2+\lambda} & \end{array}\right]$$

when σ_i is large, $\frac{\sigma_i}{\sigma_i^2+\lambda} \approx 1/\sigma_i$

(like in least squares).

when σ_i is small, $\frac{\sigma_i}{\sigma_i^2+\lambda} \approx 0$

(so we don't make noise explode)

Now note $(\Sigma^T\Sigma + \lambda I)^{-1}\Sigma^T = \Sigma^T(\Sigma\Sigma^T + \lambda I)^{-1}$

i.e. $n < p$, $\underset{(n\times p)}{\Sigma} = \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & 0 \\ & & & \sigma_n \end{array}\right]$ ($p-n$ cols)

$$\underbrace{\Sigma^T\Sigma}_{p\times p} = \begin{bmatrix} \sigma_1^2 & & & & \\ & \ddots & & & \\ & & \sigma_n^2 & & \\ & & & 0 & \\ & & & & \ddots \; 0 \end{bmatrix} \Rightarrow \underbrace{(\Sigma^T\Sigma + \lambda I)^{-1}}_{p\times p}\underbrace{\Sigma^T}_{p\times n} = \left[\begin{array}{ccc} \frac{\sigma_1}{\sigma_1^2+\lambda} & & \\ & \ddots & \\ & & \frac{\sigma_n}{\sigma_n^2+\lambda} \\ \hline & 0 & \end{array}\right]$$

$$\underbrace{\Sigma\Sigma^T}_{n\times n} + \lambda I = \begin{bmatrix} \sigma_1^2+\lambda & & \\ & \ddots & \\ & & \sigma_n^2+\lambda \end{bmatrix} \quad \Rightarrow \Sigma^T(\Sigma\Sigma^T + \lambda I)^{-1} = \left[\begin{array}{ccc} \frac{\sigma_1}{\sigma_1^2+\lambda} & & \\ & \ddots & \\ & & \frac{\sigma_n}{\sigma_n^2+\lambda} \\ \hline & 0 & \end{array}\right]$$

(similar for $p < n$)

Then $\hat{\underline{w}} = \underbrace{V\Sigma^T U^T}_{X^T} \underbrace{U(\Sigma\Sigma^T + \lambda I)^{-1} U^T \underline{y}}_{\underline{\alpha}}$

$$\Rightarrow \boxed{\hat{\underline{w}} = X^T \underline{\alpha} \text{ where } \underline{\alpha} = (U\Sigma\Sigma^TU^T + \lambda UU^T)^{-1}\underline{y} = (XX^T + \lambda I)^{-1}\underline{y}} \quad (*)$$

$\Rightarrow$ that is, $\hat{w}$ is a weighted sum of the columns of X^T

= weighted sum of the training samples

$\Rightarrow$ we can write $\hat{w} = X^T \underline{\alpha}$ where $\underline{\alpha} \in \mathbb{R}^n$ tells us how much weight is given to each training sample

Now let's compute a ridge regression estimate in "feature space"

Define $\underline{\Phi} = \begin{bmatrix} — \phi(\underline{x}_1)^T — \\ — \phi(\underline{x}_2)^T — \\ \vdots \\ — \phi(\underline{x}_n)^T — \end{bmatrix} \in \mathbb{R}^{n \times d}$

$$\hat{\underline{w}} = \arg\min_{\underline{w}} \|y - \underline{\Phi}\,\underline{w}\|_2^2 + \lambda \|\underline{w}\|_2^2$$

$$= \underline{\Phi}^T \underline{\alpha} \quad \text{where} \quad \underline{\alpha} = (\underline{\Phi}\,\underline{\Phi}^T + \lambda I)^{-1} y \quad \text{from } \circledast$$

Define $K = \underline{\Phi}\,\underline{\Phi}^T$ = Kernel matrix. Then $K_{ij} = \phi(\underline{x}_i)^T \phi(\underline{x}_j)$. This is often treated like a measure of similarity between samples $\underline{x}_i$ and $\underline{x}_j$ and can be computed directly using a "Kernel function"

$K(\underline{x}_i, \underline{x}_j) := \phi(\underline{x}_i)^T \phi(\underline{x}_j)$ — bypassing any direct computation of $\phi(\underline{x}_i)$

Then $\underline{\alpha} = (K + \lambda I)^{-1} y$ and for a new sample $\underline{x}$, the predicted label is

$$\hat{y} = \hat{\underline{w}}^T \phi(\underline{x}) = \underline{\alpha}^T \underline{\Phi}\, \phi(x) = \sum_{i=1}^{n} \alpha_i\, \phi(x_i)^T \phi(x) = \sum_{i=1}^{n} \alpha_i\, K(x_i, x)$$

Kernel Ridge Regression

We seek predictions of the from $\hat{y} = \phi(\underline{x})^T \hat{\underline{w}}_R$ where

$$\hat{\underline{w}}_R = \underset{\underline{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - \phi(\underline{x}_i)^T \underline{w}\right)^2 + \lambda \|\underline{w}\|_2^2$$

$$= \underline{\Phi}\,\underline{\alpha} \quad \text{where} \quad \underline{\alpha} = \left(\underline{\Phi}\,\underline{\Phi}^T + \lambda I\right)^{-1} y$$

We can compute such predictions efficiently via:

① $\underline{\alpha} = \left(K + \lambda I\right)^{-1} y$

② $\hat{\underline{y}} = \sum_{i=1}^{n} \alpha_i \, k(\underline{x}_i, \underline{x})$

Kernel functions

e.g. $\underline{x} \in \mathbb{R}^2$, $\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\phi(\underline{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2 \end{bmatrix}$

$K_{ij} := \phi(x_i)^T \phi(x_j) \iff K = \Phi\Phi^T$

by definition — matrix form

$\Rightarrow \phi(\underline{x}_i)^T \phi(\underline{x}_j) = x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{i2} x_{j1} x_{j2} + x_{i2}^2 x_{j2}^2$ ← Computing this directly takes $O(d)$ time where $\phi(\underline{x}) \in \mathbb{R}^d$

$= (x_{i1} x_{j1} + x_{i2} x_{j2})^2$

$= (\underline{x}_i^T \underline{x}_j)^2$ ← Computing this takes $O(p)$ time. recall $p \ll d$.

Popular Kernels

- polynomials of degree q: $K(\underline{x}_i, \underline{x}_j) = (\underline{x}_i^T \underline{x}_j)^q$ $(d = O(p^q))$
- polynomials of degree up to q: $K(\underline{x}_i, \underline{x}_j) = (\underline{x}_i^T \underline{x}_j + 1)^q$ $(d = O(p^q))$
- Gaussian Kernels: $K(\underline{x}_i, \underline{x}_j) = \exp\left(-\frac{\|\underline{x}_i - \underline{x}_j\|_2^2}{2\sigma^2}\right)$, σ^2 is a bandwidth/tuning parameter, $d = \infty$
  no closed-form expression for $\phi(\underline{x})$

Note: Kernel regression gives a linear boundary in the high-dimensional feature space

$\iff$ nonlinear boundary in original space.