Lec 16

Thursday, November 7, 2019 11:10

Recap: Augmenting the data

$$x \mapsto \varphi(x)$$

& then plug in to SVC

called this "augmented SVC"

$$\min_{\beta, \beta_0} \ \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^n \xi_i$$

$$\xi_i \geq 0 \qquad \xi_i \geq 1 - y_i(\beta^T \varphi(x_i) + \beta_0)$$

Last time rephrased as:

$$\min_{\substack{\alpha \in \mathbb{R}^n \\ \beta_0}} \ \frac{1}{2}\alpha^T \underline{K} \alpha + C\sum_{i=1}^n \xi_i$$

$$\xi_i \geq 0 \qquad \xi_i \geq 1 - y_i(\underline{K}_i^T \alpha + \beta_0)$$

Where $\underline{K}_{ij} = \underbrace{\varphi(x_i)^T \varphi(x_j)}_{K(x_i, x_j)}$ kernel fn

Conclusion: the augmented SVC only depends on the example feature data via the kernel Gram matrix $\underline{K}$

Idea: Instead of defining φ, define kernel K

Ex: Linear kernel: $K(x, x') = x^T x'$ → gives the original

$\varphi(x) = x$ SVC

Ex: Polynomial kernel

First consider $p=1$

$$K(x, x') = (1 + x x')^d$$

E.g. for $d=3$

$$K(x, x') = 1 + 3\, x \cdot x' + 3\,(x^2)(x'^2) + (x)^3 (x')^3$$

Q: What φ gives $K(x, x') = \varphi(x)^T \varphi(x')$?

$$\varphi(x) = \begin{pmatrix} 1 \\ \sqrt{3}\, x \\ \sqrt{3}\, x^2 \\ x^3 \end{pmatrix}$$

For general d:

$$K(x, x') = 1 + d\, x x' + \cdots\cdots + d\,(x x')^{d-1} + (x x')^d$$

$$\varphi(x) = (1, \sqrt{d}\, x, \cdots, \sqrt{d}\, x^{d-1}, x^d)$$

For general d, p:

$$K(x, x') = (1 + x^T x')^d$$

$\varphi(x) =$ vector of all monomials of $(x_1, \cdots, x_p)$ of degree $\leq d$

e.g. $\underbrace{x_1 x_2 x_3^2}_{\text{monomial of degree 4}}$

Let's get crazy:

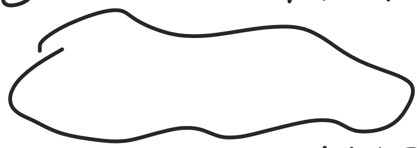
$$K(x, x') = \exp(-\|x - x'\|_2^2 / \sigma^2)$$

Radial Basis Function (RBF) kernel

Corresponding φ sends x to ~~the~~ space of functions (infinite dim)

We can even do this with unstructured data.

e.g. $X =$ string of "A, C, G, T"

$\varphi(x) =$ 

s.t. $k(x, x') = e^{-\text{editdist}(x, x')^2/\sigma^2}$

Note: Not everything is a kernel

Need:
1. $k(x, x') = k(x', x)$ symmetric
2. $k(x, x') \geq 0$
3. PSD

   $\forall x_1, \dots, x_n$

   $K_{ij} = k(x_i, x_j) \Rightarrow K$ is PSD

If we have 1-3 then $\exists \varphi$

s.t. $k(x, x') = \varphi(x)^T \varphi(x')$

Key insight: don't even need to know φ

Kernel trick generalizes to other problems

Kernel Ridge Regression

Recall ridge regression

$$\min_{\beta, \beta_0} \sum_i (y_i - \beta^T x_i - \beta_0)^2 + \lambda \|\beta\|_2^2$$

$\xrightarrow{\text{augment}} \sum_i (y_i - \beta^T \varphi(x_i) - \beta_0)^2 + \lambda \|\beta\|^2$

[illegible]

$\longrightarrow$ similar arguments as before (representer theorem + kernel trick) yield that this also just depends on the data via the kernel matrix

Kernel PCA

Recall PCA

Amounts to finding the eigendecomp of

$$[\underline{X}\,\underline{X}^T]_{ij} = x_i^T x_j$$

$$x \xrightarrow[\text{linear mapping}]{} \text{lower dim space that explains most of } x\text{'s variance}$$

Kernelize this

$$\underbrace{x \xrightarrow[\varphi]{} \text{hi-dim (possibly } \infty\text{-dim) space} \xrightarrow[\text{linear projection}]{} \text{low-dim space that explains most of variance in } \varphi}_{\text{nonlinear mapping into low-dim space that explains most of } x\text{'s variance}}$$

Apply eigendecomp to the kernel gram matrix $\underline{K}_{ij} = k(x_i, x_j)$

Centered version: apply the eigendecomp

on the centered kernel gram matrix

$$\mathbb{K}^c = \mathbb{K} - \frac{\mathbb{1}_{n\times n}}{n}\mathbb{K} - \mathbb{K}\frac{\mathbb{1}_{n\times n}}{n} + \mathbb{1}_{n\times n}\mathbb{K}\mathbb{1}_{n\times n}$$

$\leftarrow$ $n\times n$ matrix of all 1s

Linear PCA: find the q linear fns s.t.

$$z = (u_1^T x, u_2^T x, \dots, u_q^T x) \leftarrow \text{PC's}$$

explains the most variance

Kernel PCA: find the q fns $f_j \in \mathcal{H} = \text{span}(\{K(x, \cdot) : x \in \mathbb{R}$

s.t.

$$z = (f_1(x), f_2(x), \dots, f_q(x)) \leftarrow \text{kernel PC's}$$

explains the most ~~the~~ variance

Neural Networks

The "Vanilla" neural network

Add an "embedding step" to the linear model

$$x \longrightarrow z = \varphi(x) \longrightarrow \hat{f}(x) = \beta^T z$$

big diff for NN:
<u>also learn φ</u>

What if $z \in \mathbb{R}^q$ $\quad z = Ax \quad A \in \mathbb{R}^{q\times p}$

$$\hat{f}(x) = \beta^T z(x) = \beta^T A x = \underbrace{(A^T\beta)'}_{\text{still a linear model}} x$$

So the NN approach:

apply an activation/nonlinearity.

$z_j = \sigma(A_j^T x)$ where σ is <u>non</u>linear

e.g. $\sigma(u) = \frac{1}{1+e^{-u}}$

Now $\beta^T z(x)$ is <u>not</u> linear in x