INTRODUCTION TO MACHINE LEARNING COMPSCI 4ML3

Lecture 7

Hassan Ashtiani

CALCULATING OLS WITH FEATURE MAPS

$X_{n \times d_1} \longrightarrow \Phi_{n \times d_2}$

- Feature map: $\phi(x): \mathrm{R}^{d_1} \longmapsto \mathrm{R}^{d_2}$
- To calculate: $\underline{\mathrm{W}^*} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T Y \longleftarrow$
  - need to invert a $d_2 \times d_2$ matrix
  - d_2 can be very large, and even infinite!
- Kernel Trick: Compute the high-dimensional inner product efficiently
  - $K(x^i, x^j) = < \phi(x^i), \phi(x^j) >$
- Use this as a building-block for performing other operations
- Rewrite the least squares solution so that it only uses inner products in the feature map!

ROADMAP

- Optimal OLS $W^* = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T Y$
- Assume d_2 is very large, even $d_2 >> n$
- Instead of finding w, try to introduce new parameter a whose size is n rather than d_2
- Now we have n parameters
- Find optimal a as a function of K

KERNELIZED LEAST SQUARES

- $W^* = \min_W \|\Phi W - Y\|_2^2 + \lambda \|W\|_2^2$
- STEP 1: SHOW THERE EXISTS $a \in R^n$, SUCH THAT $W^* = \Phi^T a$
  - IN OTHER WORDS, $\boxed{W^* = \sum a_i \phi(x^i)}$
  - NUMBER OF PARAMETERS?
    - n INSTEAD OF d_2 …
- PROOF?

$\phi(x^i) \in \mathbb{R}^{d_2}$

$W^* =$

$$0 = \frac{\partial}{\partial w}\left(\|\Phi w - Y\|_2^2 + \lambda \|w\|_2^2 \right)$$

$$= 2w^T(\Phi^T \Phi) - 2Y^T\Phi + 2\lambda w^T = 0$$

$$\Longrightarrow \lambda w = \Phi^T Y - \Phi^T \Phi w$$

$$\Longrightarrow w = \Phi^T \underbrace{\left(\frac{Y - \Phi w}{\lambda} \right)}_{a} = \sum a_i \phi(x^i)$$

$$\Phi$$

$$(AB)^T = B^T A^T, \quad (A+B)^T = A^T + B^T, \ldots$$

KERNEL FUNCTION NOTATIONS

- $k(x^i, x^j) =< \phi(x^i), \phi(x^j) >$
- Kernel or Gram matrix of a data set:
  - $\boxed{K_{n\times n}} = [k(x^i, x^j)] = \Phi\Phi^T$

- $\boxed{k(x)} = \Phi\phi(x) = [k(x, x^1) \;\; k(x, x^2) \;\; .... \;\; k(x, x^n)]^T$

PREDICTION, GIVEN a

nice when $d_2 \gg n$

- $W^* = \Phi^T a$
- Prediction on training points
  - $\hat{Y} = \Phi W^* = ?$ $\quad \Phi \Phi^T a = \underset{n \times n}{K} \cdot a_{n \times 1}$
- Prediction for new test point x:
  - $\hat{y}(x) = < \phi(x), W^* > = ?$ $\quad \langle \phi(x), \Phi^T a \rangle$

$= a^T \underbrace{\Phi \phi(x)} = a^T K(x) = K(x)^T a$

$= \langle a, K(x) \rangle$

$K(x) = [K(x, x^1), K(x, x^2), \ldots]$

FINDING a USING DUAL FORM

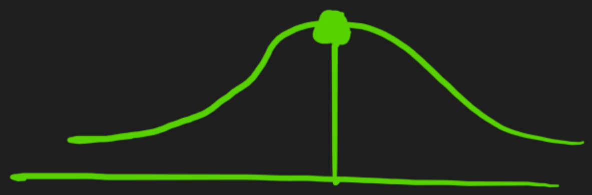
- $W^* = \min_{W} \|\Phi W - Y\|_2^2 + \lambda \|W\|_2^2$
- STEP 2: USE $W^* = \Phi^T a$ TO REFORMULATE THE PROBLEM IN TERMS OF FINDING a (DUAL FORM)
- $\min_{\vec{a} \in R^n} \|\Phi \Phi^{\mathrm{T}} a - Y\|_2^2 + \lambda \|\Phi^{\mathrm{T}} \mathbf{a}\|_2^2$ OR…

$$\min_{\vec{a}} \|Ka - Y\|_2^2 + \lambda a^T K a$$

  - $a^* = (K + \lambda I)^{-1} Y$ (PROOF?)
- FASTER WHEN $d_2 \gg n$

Choice of Kernel

- Kernel encodes similarity of points x^i and x^j
- Polynomial: $k(x, z) = (1 + x^T z)^M$
- Gaussian: $k(x, z) = e^{-\left(\frac{1}{2\sigma^2}\right)\|x-z\|_2^2} = e^{-\alpha\|x-z\|_2^2}$
  - $\phi(x)$ is infinite dimensional
- How to choose a kernel?
  - It should be valid (there must exist a ϕ)
  - Domain Knowledge
  - Kernel function captures "similarity" between points



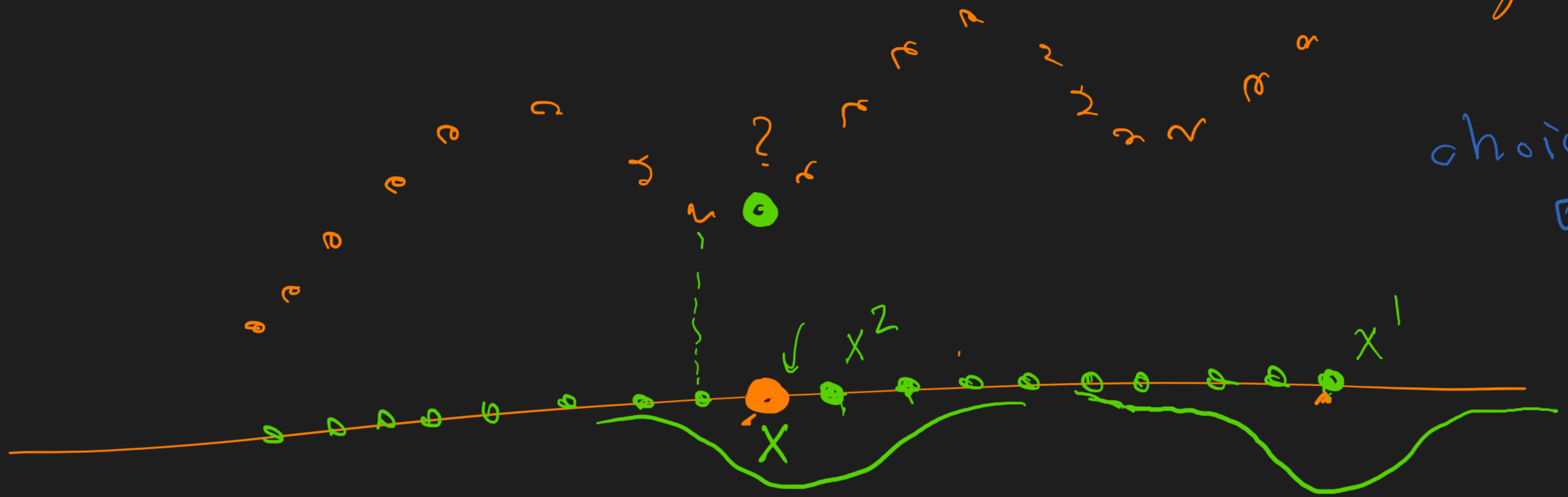
GAUSSIAN KERNEL: INTUITION

- JUPYTER NOTEBOOK

$x \in \mathbb{R}$

$y \in \mathbb{R}$

choice of σ



$\rightarrow \sum \alpha_i K(x, x^i)$, $K(x, x^1) \approx 0$

$K(x, x^2) > 0$

COMPUTATIONAL COMPLEXITY

- MATRIX MULTIPLICATION (N-BY-N MATRICES)
  - NATIVE METHOD: $O(N^3)$
  - STRASSEN'S ALGORITHM: $O(N^{2.8074})$
  - CURRENTLY BEST KNOWN METHOD: COPPERSMITH–WINOGRAD ALGORITHM $O(N^{2.3755})$
- MATRIX INVERSION
  - GAUSSIAN ELIMINATION: $O(N^3)$
  - POSSIBLE TO REDUCE IT TO MULTIPLICATION (SO $O(N^{2.3755})$)

COMPUTATIONAL COMPLEXITY

- Training complexity (n training points)
  - Regularized Least Squares
    - $W = (X^TX + \boldsymbol{\lambda I})^{-1}X^TY$
  - Kernel Least Squares
    - $\boldsymbol{a} = (\boldsymbol{K} + \boldsymbol{\lambda I})^{-1}\boldsymbol{Y}$
- Test complexity (for a single test point)
  - Regularized Least Squares
    - x^TW
  - Kernel Least Squares
    - $k(x)^Ta$

$d_2 \times d_2$ matrix

$n \times n$ matrix

$\rightarrow$ d_2 computation

$\rightarrow$ n computations