Feature maps and kernels

COMS 4771 Fall 2025

Upgrading linear models

Upgrade linear models by being creative about features

- (Where do numerical features really come from anyway?)

- Example: text data
  - One feature per word: but what numerical value to assign?

  - Stemming: map words with the same "stem" to the same canonical form

  - Stop word filtering: Ignore words like "the", "a", etc.

- Not specific to linear models

Suppose you already have numerical features $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ ...

- Instead of using x directly in linear model, can use $\varphi(x)$ for some feature map

$$\varphi \colon \mathbb{R}^d \to \mathbb{R}^p$$

  (with p possibly different, perhaps larger, than d)

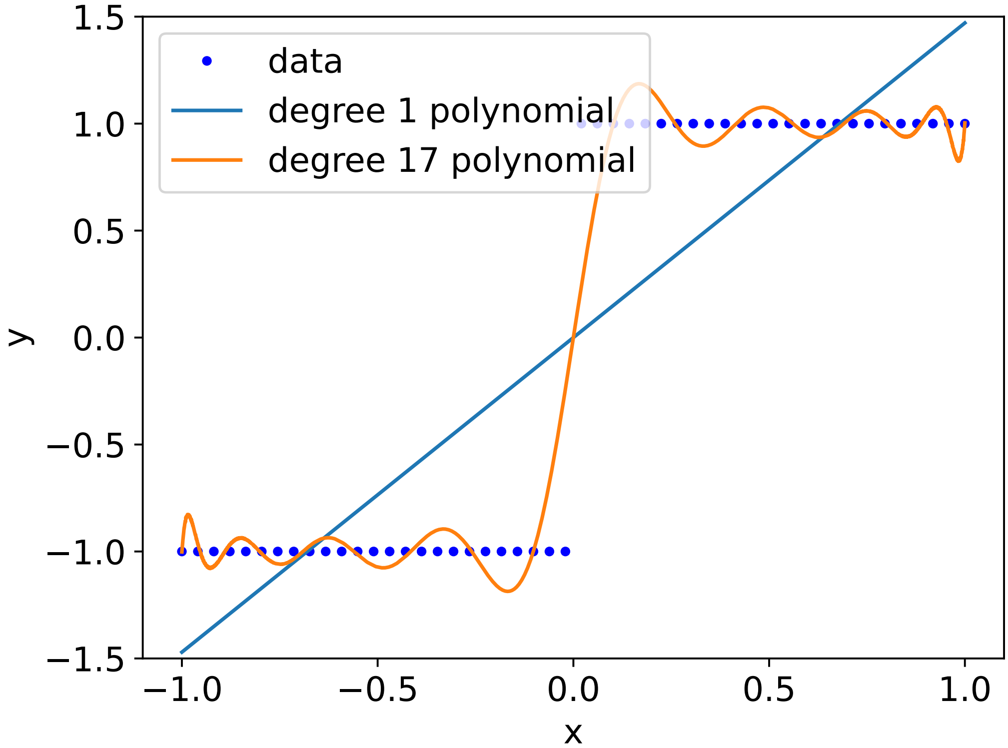
- Feature space (corresponding to φ): image of φ

Any **univariate polynomial** in x of degree $\leq k$ can be written as

$$w^\intercal \varphi(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_k x^k$$

where feature map $\varphi\colon \mathbb{R} \to \mathbb{R}^{k+1}$ is given by

$$\varphi(x) = (1, x, x^2, \ldots, x^k)$$



Any **multivariate quadratic** can be written as

$$w^\mathsf{T}\varphi(x)$$

where feature map $\varphi\colon \mathbb{R}^d \to \mathbb{R}^{1+2d+\binom{d}{2}}$ is given by

$$\varphi(x) = (1, x_1, \dots, x_d, x_1^2, \dots, x_d^2, x_1x_2, \dots, x_{d-1}x_d)$$

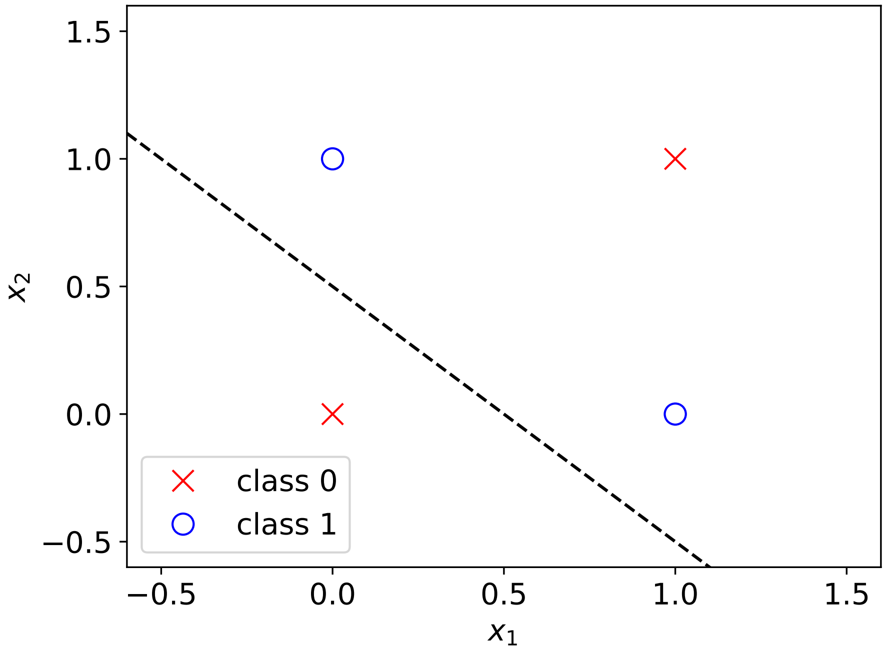
Can generalize to **arbitrary multivariate polynomials**

Using **feature maps with linear classifiers**:

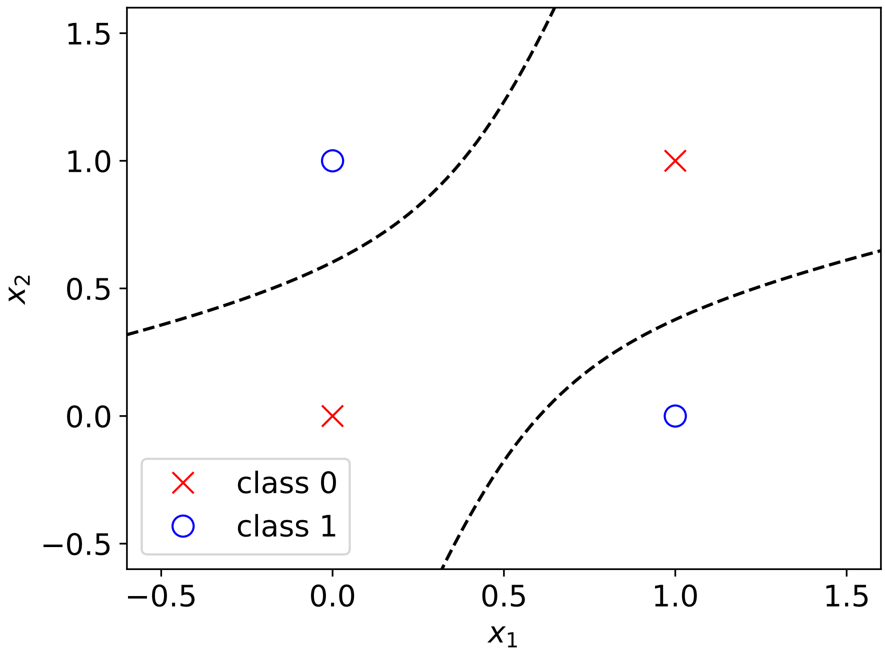
$$f_w(x) = \begin{cases} 1 & \text{if } w^\mathsf{T}\varphi(x) > 0 \\ 0 & \text{if } w^\mathsf{T}\varphi(x) \le 0 \end{cases}$$

Can get decision boundaries that are not just (affine) hyperplanes!

Not linearly separable



Using $\varphi(x) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2) \longrightarrow$ conic sections



Question: How can we choose the feature map to use?

Perceptron with feature map $\varphi\colon \mathbb{R}^d \to \mathbb{R}^p$:

- Start with $w = 0$ (p-dimensional vector)
- While there exists $(x, y) \in \mathcal{S}$ such that $f_w(x) \neq y$:
  - Let $(x, y) \in \mathcal{S}$ be any such example
  - Update w:

$$w \leftarrow \begin{cases} w + \varphi(x) & \text{if } y = 1 \\ w - \varphi(x) & \text{if } y = 0 \end{cases}$$

- Return w

Possible concern: **feature space dimension p can be large**

- Example: NIST dataset of handwritten digits
  - $d = 784$ pixels $\rightarrow p = 308505$ with quadratic feature map
- Large number of parameters
- Time to evaluate linear functions $w^\mathsf{T}\varphi(x)$ may grow with p

Kernel trick

[Kernel trick] is a way to use feature maps $\varphi\colon \mathbb{R}^d \to \mathbb{R}^p$ with linear models but avoid (explicitly) doing the following:

- represent weight vector $w \in \mathbb{R}^p$
- compute $\varphi(x)$ for any x

Only works with certain learning algorithms, called [kernel methods]:

- Main requirement: algorithm only uses feature vectors through inner products

$$\varphi(x)^\mathsf{T}\varphi(z)$$

(Variant of) quadratic feature map $\varphi\colon \mathbb{R}^d \to \mathbb{R}^{1+2d+\binom{d}{2}}$:

$$\varphi(x) = (1, \sqrt{2}x_1, \dots, \sqrt{2}x_d, x_1^2, \dots, x_d^2, \sqrt{2}x_1x_2, \dots, \sqrt{2}x_{d-1}x_d)$$

- Naïve method for computing inner product $\varphi(x)^\mathsf{T}\varphi(z)$: _____ time
  - Form $\varphi(x)$
  - Form $\varphi(z)$
  - Compute $\varphi(x)^\mathsf{T}\varphi(z)$
- Kernel trick: for any $x, z \in \mathbb{R}^d$,

$$(1 + x^\mathsf{T}z)^2 = \varphi(x)^\mathsf{T}\varphi(z)$$

  Time to evaluate: ____

(Similar trick/speed-up available for polynomial expansions of degree $k > 2$)

Kernel Perceptron

Kernel Perceptron with feature map $\varphi \colon \mathbb{R}^d \to \mathbb{R}^p$:

- Maintain "dual variable" $\alpha^{(i)}$ for each example $(x^{(i)}, y^{(i)}) \in \mathcal{S}$
- Weight vector w is implicitly represented as

$$w = \sum_i \alpha^{(i)} \varphi(x^{(i)})$$

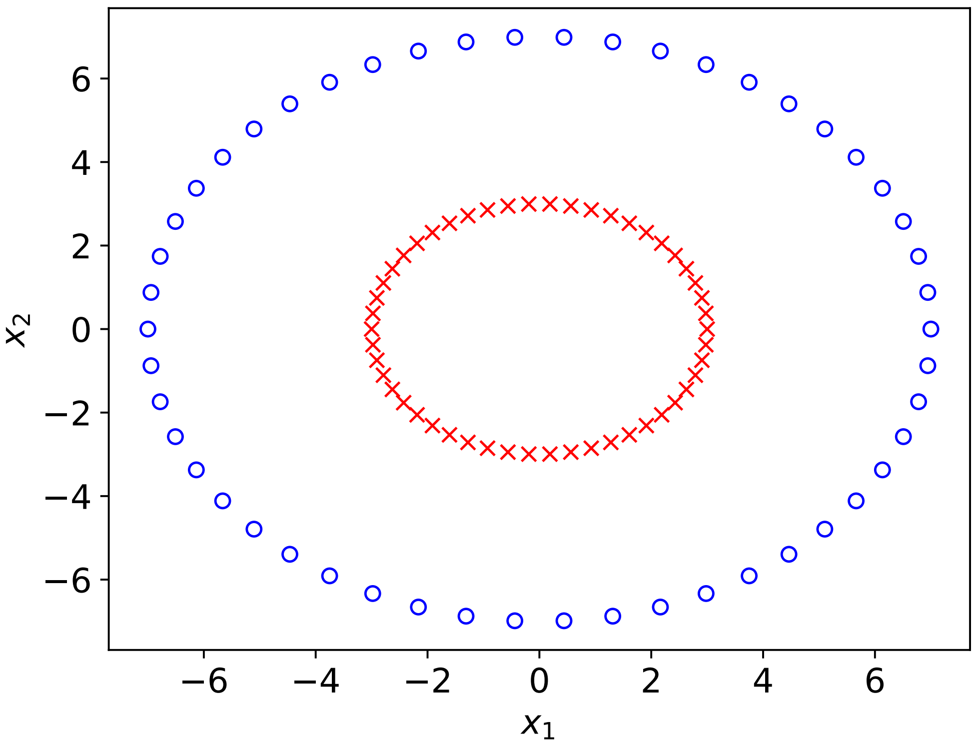
- Start with $\alpha^{(i)} = 0$ for all i
- While there exists $(x, y) \in \mathcal{S}$ such that $f_w(x) \neq y$:
  - Let $(x^{(i)}, y^{(i)}) \in \mathcal{S}$ be any such example
  - Update $\alpha^{(i)}$:

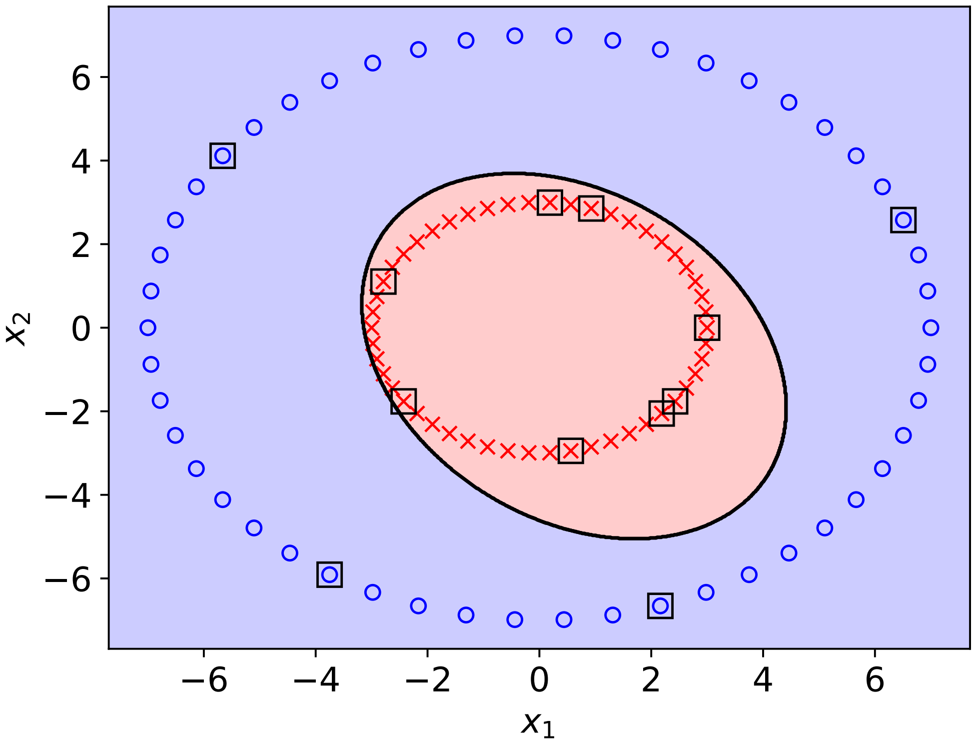
$$\alpha^{(i)} \leftarrow \begin{cases} \alpha^{(i)} + 1 & \text{if } y = 1 \\ \alpha^{(i)} - 1 & \text{if } y = 0 \end{cases}$$

- Return dual variables $(\alpha^{(i)})_{i=1}^n$

Question: What is time required to compute $f_w(x)$ in Kernel Perceptron?

(For concreteness, assume φ is the quadratic feature expansion from before)





Kernel ordinary least squares

Ordinary least squares with feature map $\varphi\colon \mathbb{R}^d \to \mathbb{R}^p$

Want to solve normal equations

$$(A^\mathsf{T} A)w = A^\mathsf{T} b$$

for $w \in \mathbb{R}^p$, but using kernel trick

$$A = \underbrace{\begin{bmatrix} \longleftarrow & \varphi(x^{(1)})^\mathsf{T} & \longrightarrow \\ & \vdots & \\ \longleftarrow & \varphi(x^{(n)})^\mathsf{T} & \longrightarrow \end{bmatrix}}_{n\times p}, \quad b = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Key fact: CS(A^T) and NS(A) are orthogonal complements

Therefore, can just look for a solution of the form $w = A^\mathsf{T}\alpha$ for some $\alpha \in \mathbb{R}^n$

$$w = A^\mathsf{T}\alpha = \sum_i \alpha^{(i)} \varphi(x^{(i)})$$

Two steps of OLS:

1. Let $\hat{b}$ be orthogonal projection of b to CS(A)

2. Solve $Aw = \hat{b}$ for w

Beyond polynomial expansions

Inner product can be regarded as "similarity function"

- E.g., text example

$$x_j = \begin{cases} 1 & \text{if article contains } j\text{-th vocabulary word} \\ 0 & \text{otherwise} \end{cases}$$

So $x^\mathsf{T} z =$ number of words the articles have in common

Kernel methods can be used with any similarity function

$$\mathrm{k} \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

as long as, for any n and any $x^{(1)}, \dots, x^{(n)} \in \mathcal{X}$, the $n \times n$ matrix

$$K = \begin{bmatrix} \mathrm{k}(x^{(1)}, x^{(1)}) & \cdots & \mathrm{k}(x^{(1)}, x^{(n)}) \\ \vdots & \ddots & \vdots \\ \mathrm{k}(x^{(n)}, x^{(1)}) & \cdots & \mathrm{k}(x^{(n)}, x^{(n)}) \end{bmatrix}$$

is positive semidefinite

(Such a similarity function is called a positive definite kernel)

Aronszajn's theorem: For any positive definite kernel $\mathrm{k}\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, there exists a feature map $\varphi\colon \mathcal{X} \to H$ such that

$$\mathrm{k}(x, z) = \varphi(x)^{\intercal}\varphi(z)$$

(H may be an infinite-dimensional vector space)

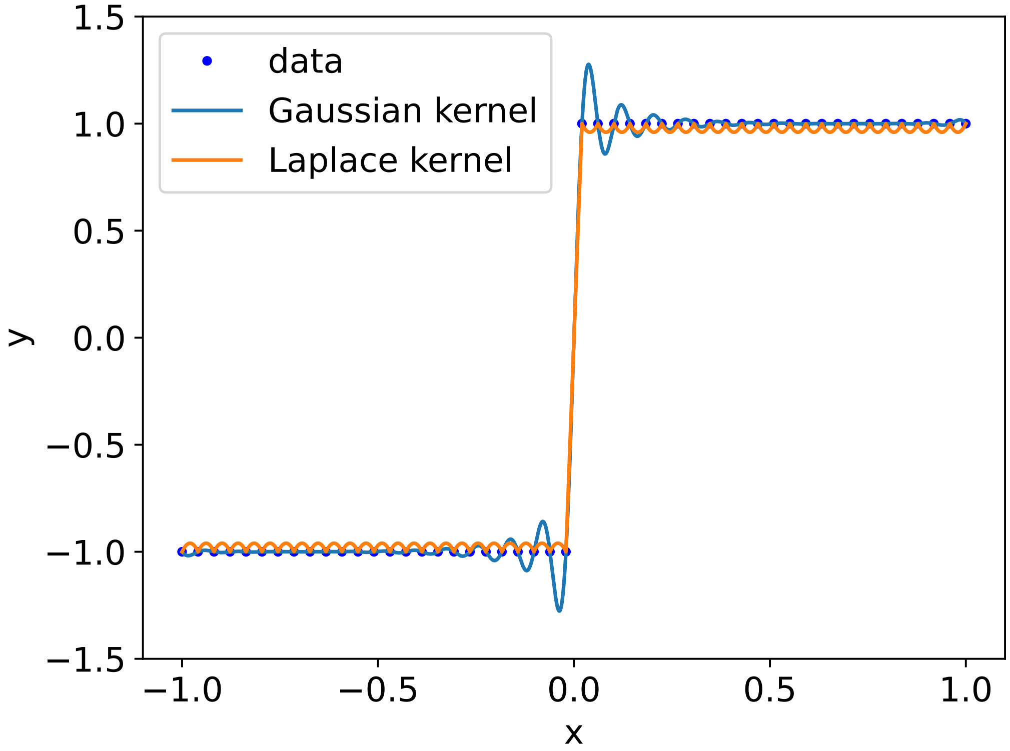
Gaussian kernel (a.k.a. radial basis function (RBF) kernel)

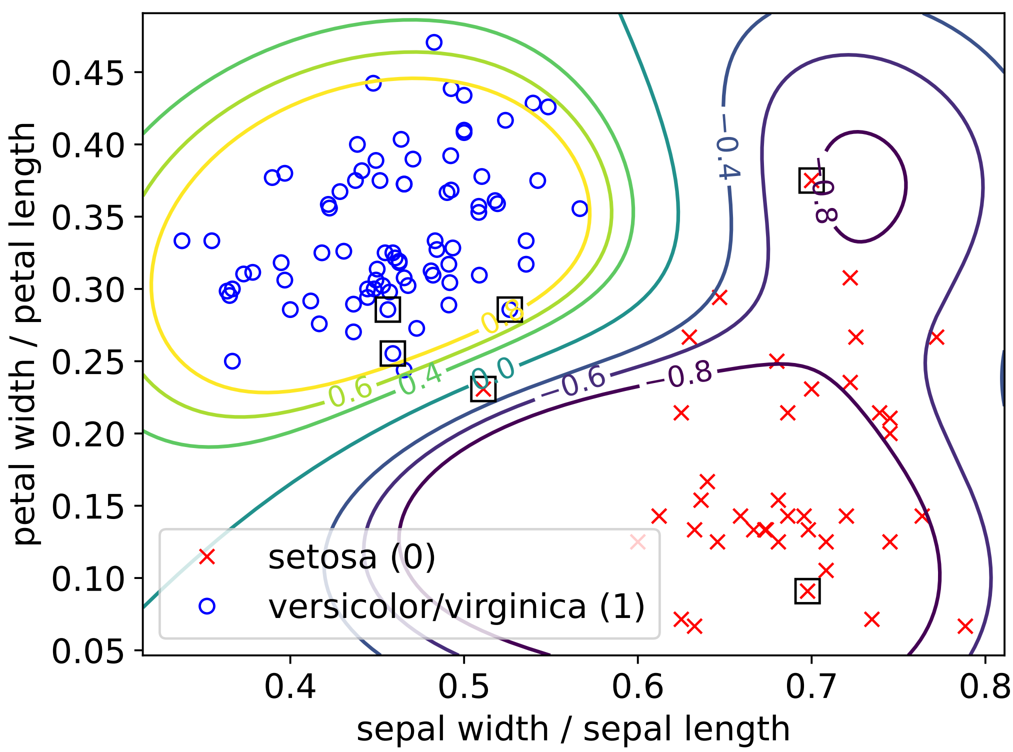
$$\mathrm{k}(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

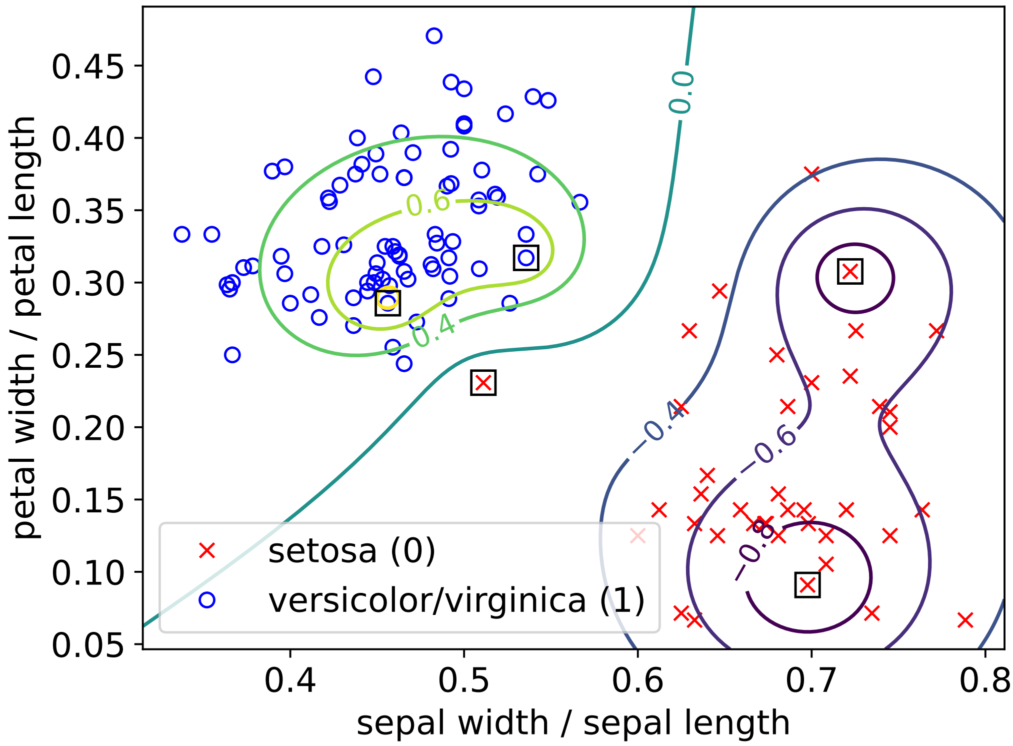
$\sigma > 0$ is bandwidth hyperparameter

Laplace kernel

$$\mathrm{k}(x, z) = \exp\left(-\frac{\|x - z\|}{\sigma}\right)$$


data
Gaussian kernel
Laplace kernel
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−1.0
−0.5
0.0
0.5
1.0
x


setosa (0)
versicolor/virginica (1)
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
petal width / petal length
0.4
0.5
0.6
0.7
0.8
sepal width / sepal length
−0.4
−0.6
−0.8
0.0
0.4
0.6



Comparison to nearest neighbors

- With Gaussian kernel, predictor is of the form

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha^{(i)} \exp\left(-\frac{\|x - x^{(i)}\|^2}{2\sigma^2}\right)$$

- What happens if x is close to $x^{(i)}$ but far from all other $x^{(j)}$, $j \neq i$?

Postscript: Are "kernel methods" are irrelevant in the age of deep learning?

- Linear models still used extensively
  - Nearest neighbor also still used extensively
- In 2021, Yale SOM professors re-discover kernelized OLS with Gaussian kernel, but poorly (i.e., they failed to figure out kernel trick, used slow approximation)
- Useful for understanding some aspects of deep learning methods