446 Section 07

Plans for today!

1. This
2. Reminders
3. Kernel Properties
4. Kernelized Linear Regression
5. PyTorch Colab Notebook

Reminders

- HW3 due next Wednesday 5/20, 11:59 PM

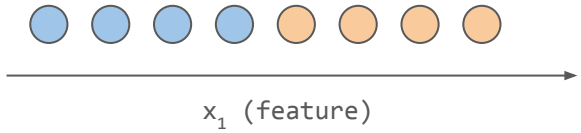
Nothing else to say… How’s life?

Kernels

Kernels ?????????????????? ??????????????

This concept stumped me when I took the class. It can be difficult to understand so spend some time digesting it

- But maybe we can simplify things… Let's start with *why*



Can you draw a horizontal line to separate these classes?

No

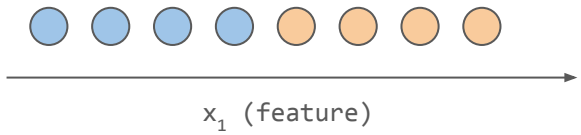
Kernels
??????????????????
??????????????

We need to create a new feature from what we have (which is x_1)

$$\phi(x) = [\phi_1(x), \phi_2(x)]$$

$$\phi_1(x) = x$$

$$\phi_2(x) = x^2$$



Can you draw a horizontal line to separate these classes?

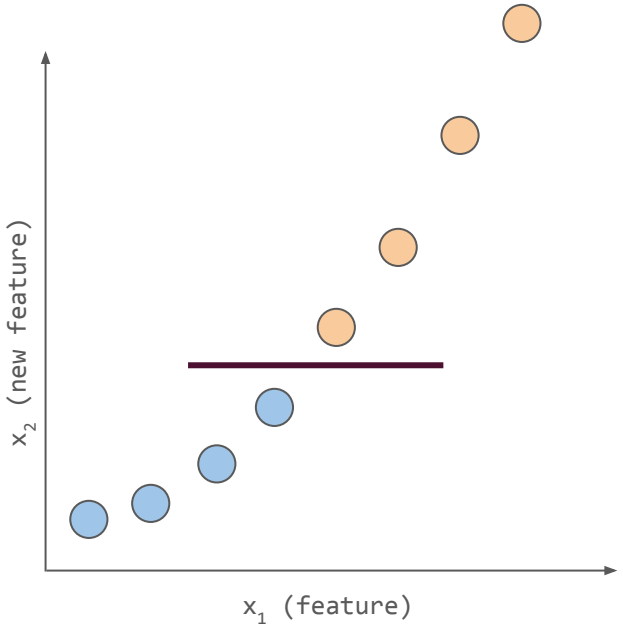
Kernels
??????????????????
??????????????

We need to create a new feature from what we have (which is x_1)

$$\phi(x) = [\phi_1(x), \phi_2(x)]$$

$$\phi_1(x) = x$$

$$\phi_2(x) = x^2$$



Can you draw a horizontal line to separate these classes?

Yes

Problems with efficiency ← THIS IS WHY!

If we did things step by step…

1. Take our *n* data points, each one a *d* dimensional vector
2. To EACH datapoint, apply our kernel map ($\phi(x)$), further blowing up the space/time complexity
3. Perform linear regression on the exploded set of features
4. Most likely fail…

Kernel trick!

Ridge Regression Reminders:

this proof isn't on your handout but feel free to follow along!

Variables:

$$X \in \mathbb{R}^{n \times d};\ y \in \mathbb{R}^{n};\ w \in \mathbb{R}^{d}$$

Optimization objective:

$$\hat{w} = \underset{w}{\operatorname{argmin}} ||y - Xw||_2^2 + \lambda ||w||_2^2$$

Closed-form solution:

$$\hat{w} = (X^\top X + \lambda I)^{-1} X^\top y$$

Our goal is to get from **ridge regression** to **kernelized linear regression**

Step 1: The Representer Theorem

The optimal weight vector $\hat{w}$ lies in the span of the data points.

$$\hat{w} = X^T \boldsymbol{\alpha} = \sum_{i=1}^{n} x_i \alpha_i$$

Where $\alpha \in \mathbb{R}^n$ is a vector of coefficients.

This fact implies that **we can write *w* as a linear combination** of the data points (x_i) and some scale factors (a_i) → useful for our proof later!

Let's do feature expansion!

From now on we are working with the *expanded* feature set:

$$X \to \phi(X)$$

NEW Optimization objective:

$$\hat{w}_\phi = \underset{w}{\text{argmin}} ||y - \phi(X)w||_2^2 + \lambda||w||_2^2$$

What if our feature set blows up so that d >> n? Or even blows up to infinite dimensions?

Step 2: Substitution

$$X \rightarrow \phi(X)$$

$$\hat{w}_\phi = \underset{w}{\operatorname{argmin}} ||y - \phi(X)[w]||_2^2 + \lambda ||[w]||_2^2$$

$$\text{If } w = \phi(X)^\top a$$

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \; \|\boldsymbol{y} - \boldsymbol{\phi(X)[\phi(X)^T \alpha]}\|_2^2 + \lambda \|[\boldsymbol{\phi(X)^T \alpha}]\|_2^2$$

This is why we needed step 1. It makes this substitution possible **no matter the dimensionality** of the “blown up” X!

Step 2: Define the Kernel Matrix

The Kernel Function:

$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

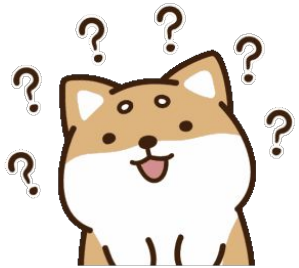
(Measures similarity between two points)

The Kernel Matrix:

$\mathbf{K} \in \mathbb{R}^{n \times n}$

where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$

How can we construct the kernel **matrix** using the kernel *function?*



Step 2: Define the Kernel Matrix

The Kernel Function:

$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

(Measures similarity between two points)

The Kernel Matrix:

$\mathbf{K} \in \mathbb{R}^{n \times n}$

where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$

$$\mathbf{K} \begin{bmatrix} \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_1) & \cdots & \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_1) & \cdots & \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_n) \end{bmatrix}$$

The Trick: We can compute **K** without ever calculating ϕ.

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

$$K_{ij} \in \mathbb{R} \quad , \quad K_{ij} = K(x_i, x_j)$$

$$K = \begin{matrix} \\ n \end{matrix} \overbrace{\begin{bmatrix} \phi(x_1)^T\phi(x_1) & \phi(x_2)^T\phi(x_1) & \cdots & \phi(x_n)^T\phi(x_1) \\ \phi(x_1)^T\phi(x_2) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \phi(x_1)^T\phi(x_n) & \cdots & \cdots & \phi(x_n)^T\phi(x_n) \end{bmatrix}}^{n}$$

The main takeaway is the formulation of the kernel matrix below.

$$n \overbrace{\begin{bmatrix} — & \phi(x_1)^T & — \\ & \vdots & \\ — & \phi(x_n)^T & — \end{bmatrix}}^{d} \times d \overbrace{\begin{bmatrix} | & & | \\ \phi(x_1) & \cdots & \phi(x_n) \\ | & & | \end{bmatrix}}^{n}$$

Where d can be infinite

$$K = \Phi(X)\Phi(X)^T \quad : \quad X \in \mathbb{R}^{n \times m}$$

Step 4: Substitute into Linear Regression

$$\text{If } \mathbf{K} = \phi(X)\phi(X)^\top$$

$$\begin{aligned}
\hat{a} &= \underset{a}{\operatorname{argmin}} ||y - \phi(X)\phi(X)^\top a||_2^2 + \lambda ||\phi(X)^\top a||_2^2 \\
&= \underset{a}{\operatorname{argmin}} ||y - \phi(X)\phi(X)^\top a||_2^2 + \lambda a^\top \phi(X)\phi(X)^\top a \\
&= \underset{a}{\operatorname{argmin}} ||y - \mathbf{K}a||_2^2 + \lambda a^\top \mathbf{K} a
\end{aligned}$$

The dimension d has completely vanished.
The problem now depends only on n.

Step 5: Derive

2a. Kernelized Linear Regression

Recall that the definition of a kernel is the following:
Definition 1. A function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a *kernel* for a map ϕ if $K(x, x') = \phi(x) \cdot \phi(x') = \langle \phi(x), \phi(x') \rangle$ for all x, x'.

Consider regularized linear regression (without a bias, for simplicity). Our objective to find the optimal parameters $\hat{w} = \arg\min_w L(w)$ for a dataset $(x_i, y_i)_{i=1}^n$ that minimize the following loss function:

$$L(w) = \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda ||w||_2^2$$

Note that from class, we know there is an optimal $\hat{w}$ that lies in the span of the datapoints. Concretely, there exist $\alpha_1, ..., \alpha_n \in \mathbb{R}$ such that $\hat{w} = \sum_i^n \alpha_i x_i$. Also recall from lecture that the expression of our loss function $L(w)$ in terms of the kernel is:

$$L(w) = ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha$$

Solve for the optimal α

2a. Kernelized Linear Regression

$$L(w) = ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda\alpha^T\mathbf{K}\alpha \rightarrow \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{K}\alpha - \alpha^T\mathbf{K}^T\mathbf{y} + \alpha^T\mathbf{K}^T\mathbf{K}\alpha + \lambda\alpha^T\mathbf{K}\alpha$$

$$\rightarrow \mathbf{y}^T\mathbf{y} - 2\alpha^T\mathbf{K}\mathbf{y} + \alpha^T\mathbf{K}^2\alpha + \lambda\alpha^T\mathbf{K}\alpha$$

1. **Gradient:** Take derivative of $L(\alpha)$ w.r.t α:

$$\nabla_\alpha L = -2K(y - K\alpha) + 2\lambda K\alpha$$

2. **First-Order Optimality:** Set to 0 and factor out K:

$$K(K\alpha - y + \lambda\alpha) = 0$$

$$K((K + \lambda I)\alpha - y) = 0$$

3. **Invert:** Assuming K is positive semi-definite (PSD):

$$(K + \lambda I)\alpha = y$$

4. **Solution:**

$$\boxed{\hat{\alpha} = (K + \lambda I)^{-1} y}$$

$$\nabla_\alpha L(w) = 0$$

$$-2\mathbf{K}(\mathbf{y} - \mathbf{K}\alpha) + 2\lambda\mathbf{K}\alpha = 0$$

$$-\mathbf{K}(\mathbf{y} - \mathbf{K}\alpha) + \lambda\mathbf{K}\alpha = 0$$

$$\mathbf{K}(\mathbf{K}\alpha - \mathbf{y} + \lambda\alpha) = 0$$

$$\mathbf{K}((\mathbf{K} + \lambda I)\alpha - \mathbf{y}) = 0$$

$$\mathbf{K}(\mathbf{K} + \lambda I)\alpha = \mathbf{K}\mathbf{y}$$

$$\hat{\alpha} = (\mathbf{K} + \lambda I)^{-1}\mathbf{y}$$

Piece-by-piece explanation

A kernel $K(a, b)$ takes in d dimensional vectors a, b and gives us a scalar.

When you construct a kernel such that $K(a, b) = \phi(a)^\top \phi(b)$

We can ultimately use K to efficiently apply ϕ to our features.

Mercer's conditions:

- *K* must be symmetric
- *K* must be positive definite

Can have infinite dimensions

Step 4: Predict

Normal case: $\hat{y} = \hat{w} \cdot z \quad : \quad z \in \mathbb{R}^d$

Kernel case: $\hat{y} = \hat{w}_\phi \cdot \phi(z) \qquad \hat{w}_\phi = \phi(x)^T a$

$$= \phi(x)^T a \, \phi(z)$$

$$= \sum_{i=1}^{n} a_i \phi(x_i)^T \phi(z)$$

$$= \sum_{i=1}^{n} a_i K(x_i, z)$$

Let's vectorize this! →

You need some extra steps to predict a new datapoint, **but the predictive power compared to computational cost is well worth it!**

Question 2b

(b) Let us assume that we were using a linear kernel where $\mathbf{K}_{ij} = x_i^T x_j$. Suppose we have $\mathbf{X}_{\text{test}}$ that we want to make prediction for after training on $\mathbf{X}_{\text{train}}$. Express the estimate $\hat{\mathbf{Y}}$ in terms of $\mathbf{K}_{\text{train}} = \mathbf{X}_{\text{train}}\mathbf{X}_{\text{train}}^T$, $\mathbf{y}_{\text{train}}$, $\mathbf{X}_{\text{train}}$ and $\mathbf{X}_{\text{test}}$. What would the general prediction formula look like if we are not using a linear kernel? Express the solution in terms of $\mathbf{K}_{\text{train, test}}$

Solution:

$$\begin{aligned}\hat{\mathbf{Y}} &= \mathbf{X}_{\text{test}}\hat{w} \\ &= \mathbf{X}_{\text{test}}\mathbf{X}_{\text{Train}}^T\hat{\alpha} \\ &= \mathbf{X}_{\text{test}}\mathbf{X}_{\text{train}}^T\left(\mathbf{K}_{\text{train}} + \lambda I\right)^{-1}\mathbf{y}_{\text{train}}\end{aligned}$$

General Solution for Kernel Ridge

$$\hat{\mathbf{Y}} = \mathbf{K}_{\text{train, test}}\hat{\alpha}$$

Where $\mathbf{K}_{train,test} = \mathbf{X}_{test}\mathbf{X}_{train}^T$

RBF Kernels

$$K(x, y) = e^{-\gamma \|x - y\|^2}$$

RBF kernels measure the similarity between the input vectors by calculating their distance in the input feature space.

If x and y are close together, K(x, y) approaches 1

If x and y are further apart, K(x, y) approaches 0

RBF Kernel is infinite-dimensional

Consider a Taylor series expansion

$$e^z = \sum_{n=0}^{\infty} \frac{z^n}{n!} = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots$$

$$\boxed{z = -\gamma \|x - y\|^2}$$

Taylor series expansion of RBF Kernel

$$K(x,y) = e^{-\gamma\|x-y\|^2}$$

$$= 1 + (-\gamma\|x-y\|^2) + \frac{(-\gamma\|x-y\|^2)^2}{2!} + \frac{(-\gamma\|x-y\|^2)^3}{3!} + \dots$$

$$= 1 - \gamma\|x-y\|^2 + \frac{\gamma^2\|x-y\|^4}{2!} - \frac{\gamma^3\|x-y\|^6}{3!} + \dots$$

Feature 1

Feature 2

Feature 3

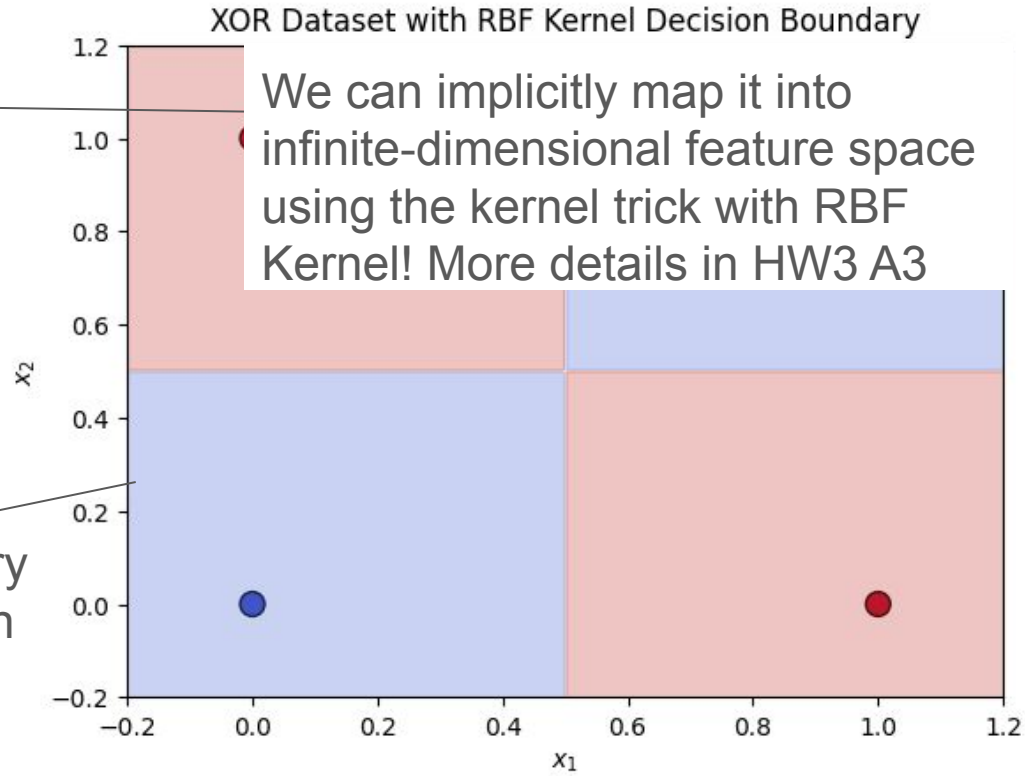
Feature 4

There's infinite number of features!

Why do we care about infinite dimensions?

By mapping the XOR dataset into infinite-dimensional feature space, we can use linear regression to draw a linear decision boundary, correctly separating the XOR dataset in this infinite-dimensional feature space

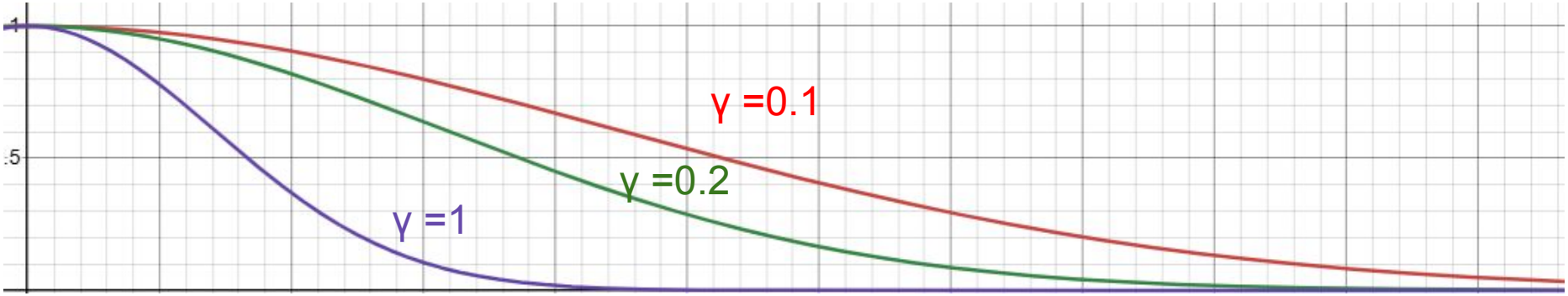
This is what the decision boundary looks like when we project it down to the original feature space

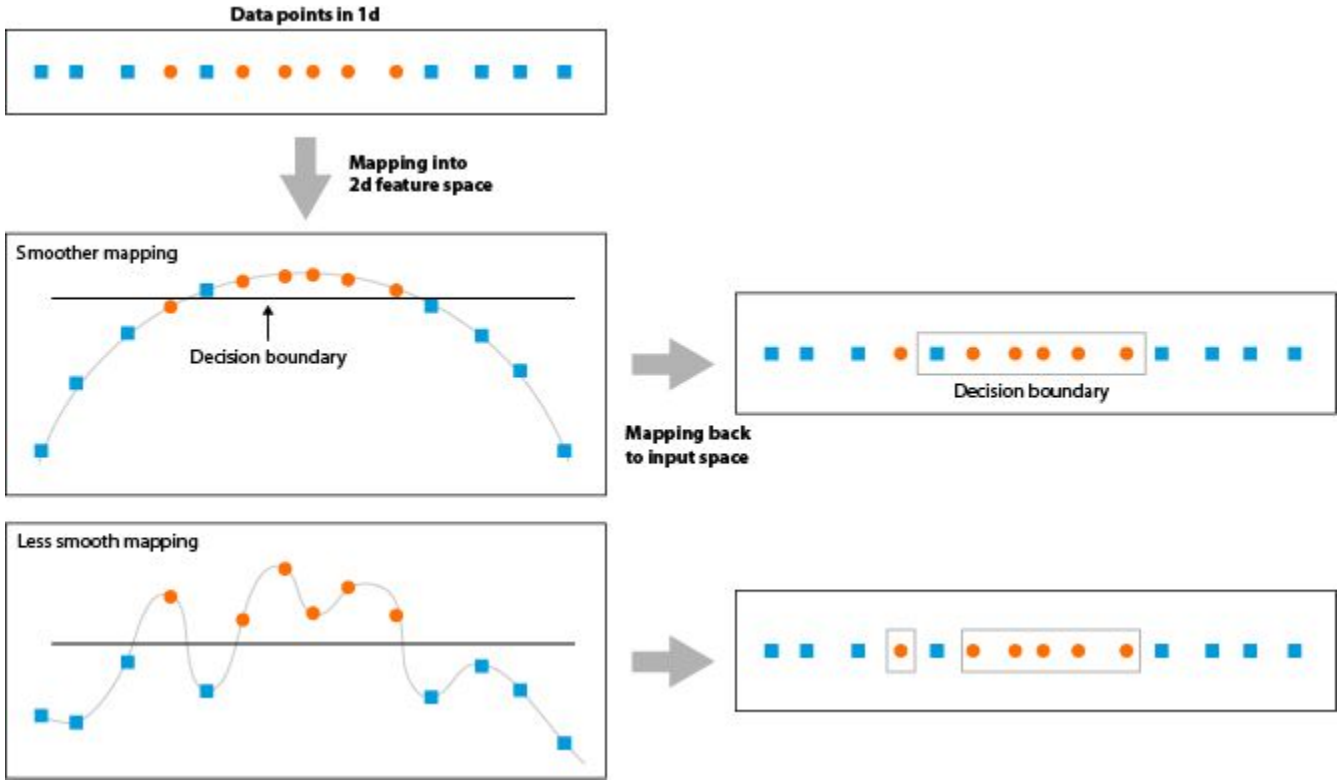


$$K(x, y) = e^{-\gamma \|x-y\|^2}$$

Controls the rate of decay for the similarity score

Smaller gamma results in a smoother mapping





A less smooth mapping can cause overfitting!

You will get more practice with using kernel regression in HW3

- $k_{\text{poly}}(x, z) = (1 + x^\top z)^d$ where $d \in \mathbb{N}$ is a hyperparameter,
- $k_{\text{rbf}}(x, z) = \exp(-\gamma \|x - z\|_2^2)$ where $\gamma > 0$ is a hyperparameter[1].

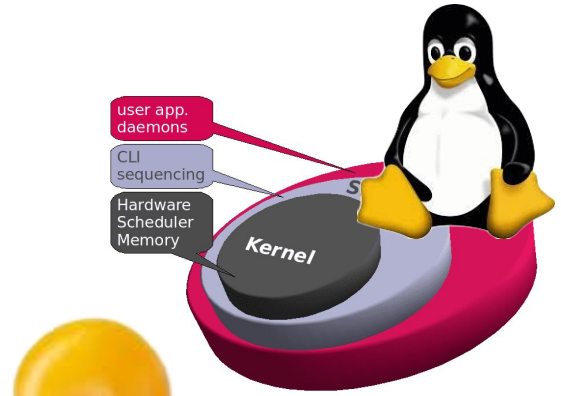
You will be doing polynomial kernel regression, and RBF kernel regression, just like what we showed you today, but on the same dataset.
You will also experiment with searching for hyperparameters such as gamma.

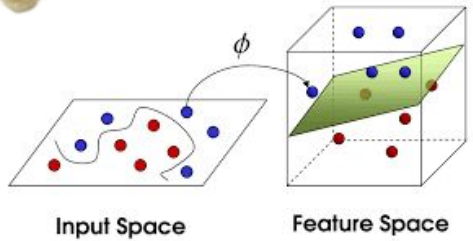
Takeaways

- Kernels are at their core a *computational efficiency trick*
  - Employed when feature mapping is computationally too much
- Kernels must satisfy Mercer's condition
  - Symmetric, positive-definite
- We perform kernelized linear regression which has extra steps

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$Ax = 0$$





What you think of when someone says "Kernel" says a lot about you…

Gradescope Section Participation Question

What is the dimensionality of the RBF kernel’s phi(x)?

Infinity

PyTorch Colab

Questions/Chat Time!