

kernel methods

SIMILARITY, FEATURE SPACES, & NON-LINEAR LEARNING

What is a kernel method?:

- ↳ Machine-learning algorithms that solve nonlinear problems by transforming them into linear problems in a higher-dimensional space $\rightarrow$ called a feature space.
- ↳ Instead of explicitly transforming the data, kernel methods only compute a function (called a kernel function) which measures similarity between 2 points.
- ↳ Examples of tasks:
 - ↳ classification
 - ↳ clustering
 - ↳ regression
 - ↳ dimensionality reduction
 - ↳ pattern recognition
- ↳ Examples of algorithms
 - ↳ Support Vector Machines (SVM)
 - ↳ Kernel PCA
 - ↳ Gaussian processes
 - ↳ Ridge regression
 - ↳ Spectral clustering

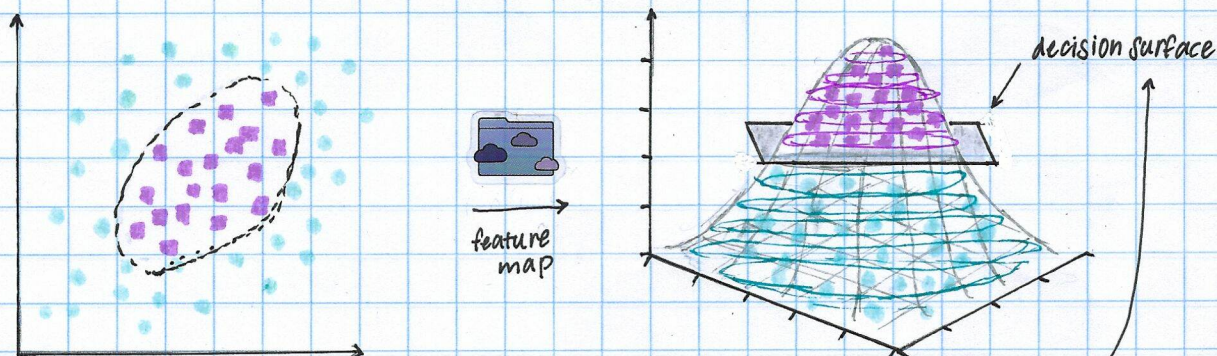
why is it useful?

the kernel trick helps bypass heavy computation and retain simple model + upgrade traditional linear tools into non-linear ones (like the ones listed)



Feature Maps:

- ↳ Suppose we have a transformation $\phi: \mathcal{X} \rightarrow \mathcal{V}$ where $\mathcal{X}$ is the original input space & $\mathcal{V}$ is the higher dimensional feature space
- ↳ Instead of working with x , we work with $\phi(x)$.



$\rightarrow$ For example, an oval or a circle might become a plane in 3D

The Kernel Function Trick:

↳ A kernel is simply a similarity measure between 2 vectors $k(x, x')$

↳ For many kernels: $k(x, x') = \phi(x)^T \phi(x')$

↳ means that the kernel computes the inner product in feature space without explicitly finding $\phi(x)$

↳ Interpretation: larger $k(x, x') \rightarrow$ points are similar, small $k(x, x') \rightarrow$ points are different

↳ so normally we transform data into higher dimensions & then compute inner products, kernel methods avoid the data transformation, compute kernel function

KEY IDEA

↳ Explicit transformation:

$$x \rightarrow \phi(x)$$

↳ Kernel trick:

$$k(x, x') = \phi(x)^T \phi(x')$$

without computing $\phi(x)$

↳ sometimes feature space is infinitely-dimensional



Kernel trick computes similarities directly

Original Data

Feature Mapping $\phi(x)$

High-Dimensional Space

Linear Separation

Kernel Classifiers & Kernel / Gram Matrix:

↳ Kernel classifiers predict labels using weighted similarities to training examples

$$\hat{y} = \text{sgn} \left(\sum_{i=1}^n w_i y_i k(x_i, x) \right) \rightarrow \text{where: } x_i = \text{training points, } y_i = \text{labels } (-1 \text{ or } +1),$$

$w_i = \text{learned weights, } k(x_i, x) = \text{similarity, } \hat{y} = \text{prediction}$

↳ Prediction depends on which training points are similar, how important they are, & what class they belong to.

↳ For n training points: $K_{ij} = k(x_i, x_j)$

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots \\ k(x_2, x_1) & k(x_2, x_2) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

★ dark =
high similarity

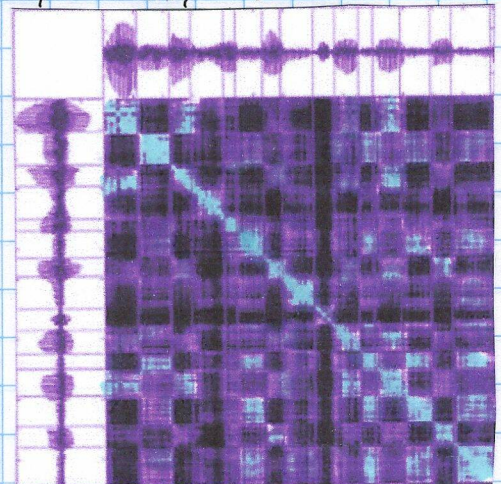
★ light =
low similarity

↳ Properties:

↳ Symmetric: $K_{ij} = K_{ji}$

↳ Positive Semi-definite for any vector v : $v^T K v \geq 0$

• Comes from Mercer's Condition



Kernel / Gram Matrix

Popular kernels:

↳ Linear kernel: $k(x, x') = x^T x'$ → simply computes the standard dot product

↳ Polynomial kernel: $k(x, x') = (x^T x' + c)^d$ → captures polynomial relationships

↳ c : constant
↳ d : degree

parameters

↳ Radial Basis Function (RBF) kernel

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \quad \rightsquigarrow \quad \|x - x'\|^2 = \text{squared Euclidean distance}$$

σ = width parameter

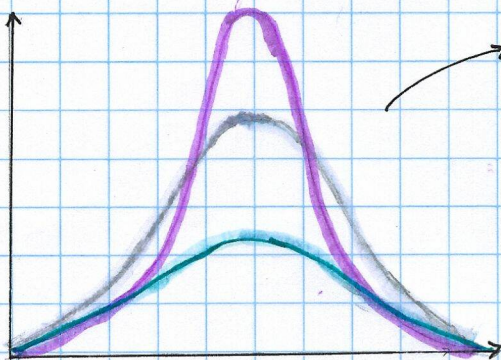
↳ most widely used kernel is SVMs (used in biofinder)

↳ kernel measures similarity based on distance.

• If $\|x - x'\| \approx 0 \rightarrow k(x, x') \approx 1$ (closer)

• If $\|x - x'\| \gg 0 \rightarrow k(x, x') \approx 0$ (further)

→ similarity decreases exponentially as distance increases



Gaussian curves

↳ small σ : narrow peak

↳ medium σ

↳ large σ : wide peak

very localized influence
complex boundaries
risk of overfitting

Smoother boundaries
more general patterns
risk of underfitting

Parameter σ

↳ RBF kernel acts like a bubble of influence

↳ each training point creates a region around itself

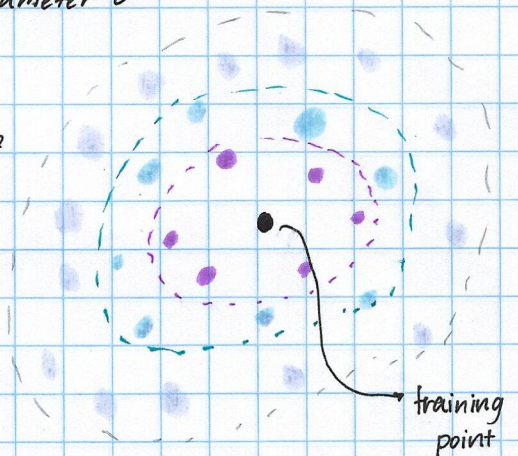
↳ nearby points receive strong influence:

$$k(x, x') \approx 1$$

↳ Faraway points receive almost no influence:

$$k(x, x') \approx 0$$

↳ The decision boundary is built by combining all these local influences



training point

Similarity (key) - $k(x, x')$

● = high

● = low

● = medium

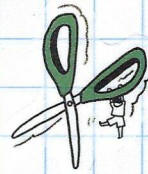
Local
Similarity
Neighborhoods

↳ Overlap of these neighborhoods allows kernel methods to separate data that is not linearly separable in the original feature space

Advantages & Disadvantages:

- ↳ handles nonlinear problems
- ↳ works in very high dimensions
- ↳ no explicit feature mapping needed
- ↳ extremely flexible
- ↳ Strong theoretical foundation
- ↳ Computationally expensive for large data sets
- ↳ Kernel Selection matters
- ↳ Hyperparameters must be tuned
- ↳ Kernel matrix grows as n^2 for n data points

advantages!



disadvantages!

Key formulas:

↳ Feature map:

$$\phi: \mathcal{X} \rightarrow \mathcal{V}$$

↳ Kernel definition:

$$k(x, x') = \phi(x)^T \phi(x')$$

↳ Kernel classifier:

$$\hat{y} = \text{sgn} \left(\sum_i w_i y_i k(x_i, x') \right)$$

↳ Gram matrix:

$$K_{ij} = k(x_i, x_j)$$

↳ RBF kernel:

$$K(x, x') = \exp \left(- \frac{\|x - x'\|^2}{2\sigma^2} \right)$$

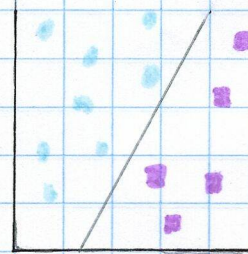
↳ Polynomial kernel

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

↳ Linear kernel:

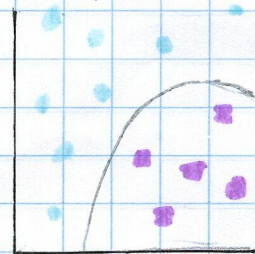
$$K(x, x') = x^T x'$$

Linear



different types
of kernels

Polynomial
 $d=2$



vs.

Polynomial
 $d=3$

