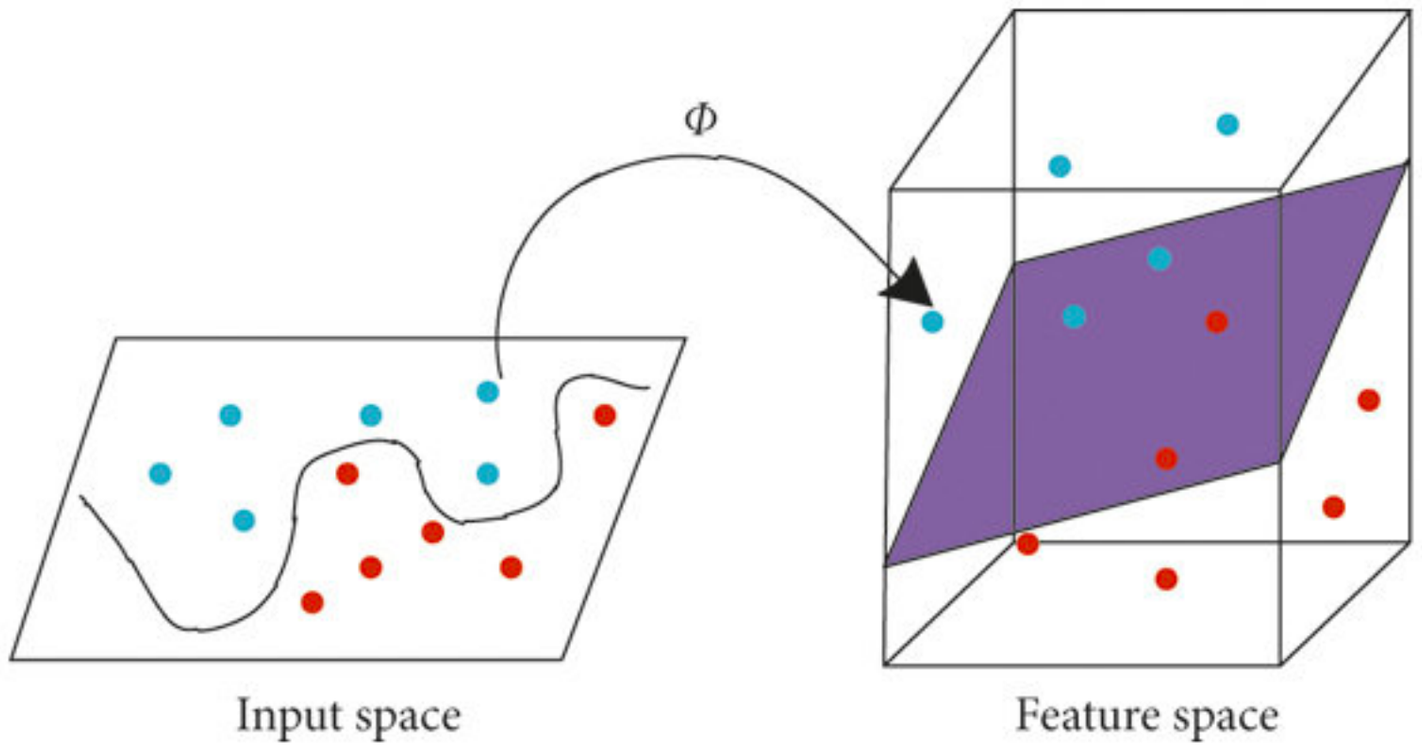


Predictions Based on Dual Formulation, and Kernel Functions

Leandro L. Minku

Overview

- Making predictions based on the dual representation
- Kernels as similarity functions
- Examples of kernel functions

Primal Formulation

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 \right\}$$

Subject to: $y^{(n)}(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}^{(n)}) + b) \geq 1$

$\forall(\mathbf{x}^{(n)}, y^{(n)}) \in \mathcal{T}$

Dual Formulation

$$\underset{\mathbf{a}}{\operatorname{argmax}}\, \tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a^{(n)} - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a^{(n)}a^{(m)}y^{(n)}y^{(m)}k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)})$$

Linear Kernel

Where: $k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) = \boldsymbol{\phi}(\mathbf{x}^{(n)})^T\boldsymbol{\phi}(\mathbf{x}^{(m)})$ e.g., $\mathbf{x}^T\mathbf{z}$

Subject to: $a^{(n)} \geq 0,\ \forall n \in \{1,\cdots,N\}$ $\sum_{n=1}^{N} a^{(n)}y^{(n)} = 0$

Primal Formulation

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 \right\}$$

Subject to: $y^{(n)}(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}^{(n)}) + b) \geq 1$

$\forall(\mathbf{x}^{(n)}, y^{(n)}) \in \mathcal{T}$

Dual Formulation

$$\underset{\mathbf{a}}{\operatorname{argmax}}\, \tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a^{(n)} - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a^{(n)}a^{(m)}y^{(n)}y^{(m)}k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)})$$

Polynomial Kernel

Where: $k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) = \boldsymbol{\phi}(\mathbf{x}^{(n)})^T\boldsymbol{\phi}(\mathbf{x}^{(m)})$ e.g., $(1 + \mathbf{x}^T\mathbf{z})^p$

Subject to: $a^{(n)} \geq 0,\ \forall n \in \{1, \cdots, N\}$ $\sum_{n=1}^{N} a^{(n)}y^{(n)} = 0$

We got rid of $\mathbf{w}$ and b. How to make predictions?

Making Predictions

Primal: $h(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + b$

- $h(\mathbf{x}) > 0 \rightarrow$ class +1
- $h(\mathbf{x}) < 0 \rightarrow$ class -1

Substituting $\mathbf{w} = \sum_{n=1}^{N} a^{(n)} y^{(n)} \boldsymbol{\phi}(\mathbf{x}^{(n)})$ (from KKT stationarity condition)

Dual: $h(\mathbf{x}) = \sum_{n=1}^{N} \overset{\geq 0}{a^{(n)}} y^{(n)} k(\mathbf{x}, \mathbf{x}^{(n)}) + b$

- $h(\mathbf{x}) > 0 \rightarrow$ class +1
- $h(\mathbf{x}) < 0 \rightarrow$ class -1

We are now making predictions on new examples based on the training examples.

Do we need to store and go through all training examples for making predictions?

Dual: $h(\mathbf{x}) = \sum_{n=1}^{N} a^{(n)} y^{(n)} k(\mathbf{x}, \mathbf{x}^{(n)}) + b$

- $h(\mathbf{x}) > 0 \rightarrow$ class +1
- $h(\mathbf{x}) < 0 \rightarrow$ class -1

- Either: $a^{(n)} = 0$, so the value of $y^{(n)} k(\mathbf{x}, \mathbf{x}^{(n)})$ won't matter.
- Or: $a^{(n)} > 0$, so this is a support vector, i.e., an example that defines the position of the decision boundary.
  - From KKT complementary slackness, $a^{(n)}(1 - y^{(n)}(\mathbf{w}^T \phi(\mathbf{x}^{(n)}) + b)) = 0$.
  - So, $(1 - y^{(n)}(\mathbf{w}^T \phi(\mathbf{x}^{(n)}) + b)) = 0$ for these examples, i.e., $y^{(n)} h(x^{(n)}) = 1$.
  - Therefore, all support vectors are on the margin.

Function h Using Only Support Vectors

$$h(\mathbf{x}) = \sum_{n=1}^{N} a^{(n)} y^{(n)} k(\mathbf{x}, \mathbf{x}^{(n)}) \quad + b$$

$$h(\mathbf{x}) = \sum_{n \in S} a^{(n)} y^{(n)} k(\mathbf{x}, \mathbf{x}^{(n)}) \quad + b$$

where S is the set of indexes of the support vectors

We only need to store the support vectors for making predictions.

Calculating b

Note that $y^{(n)}h(\mathbf{x}^{(n)}) = 1$ for each and every support vector.

So, for a given support vector $(\mathbf{x}^{(n)}, y^{(n)})$, we have that:

$y^{(n)}h(\mathbf{x}^{(n)}) = 1$ Multiply by $y^{(n)}$

$y^{(n)^2}h(\mathbf{x}^{(n)}) = y^{(n)}$ Note that $y^{(n)^2} = 1$

$h(\mathbf{x}^{(n)}) = y^{(n)}$ Substituting $h(\mathbf{x}^{(n)}) = \sum_{m \in S} a^{(m)} y^{(m)} k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) + b$

$$\sum_{m \in S} a^{(m)} y^{(m)} k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) + b = y^{(n)}$$

$$b = y^{(n)} - \sum_{m \in S} a^{(m)} y^{(m)} k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)})$$

Averaging for All Support Vectors

$$b = y^{(n)} - \sum_{m \in S} a^{(m)} y^{(m)} k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)})$$

- We have N_S support vectors.
- We can compute b for each of them and average the results to get a numerically more stable solution:

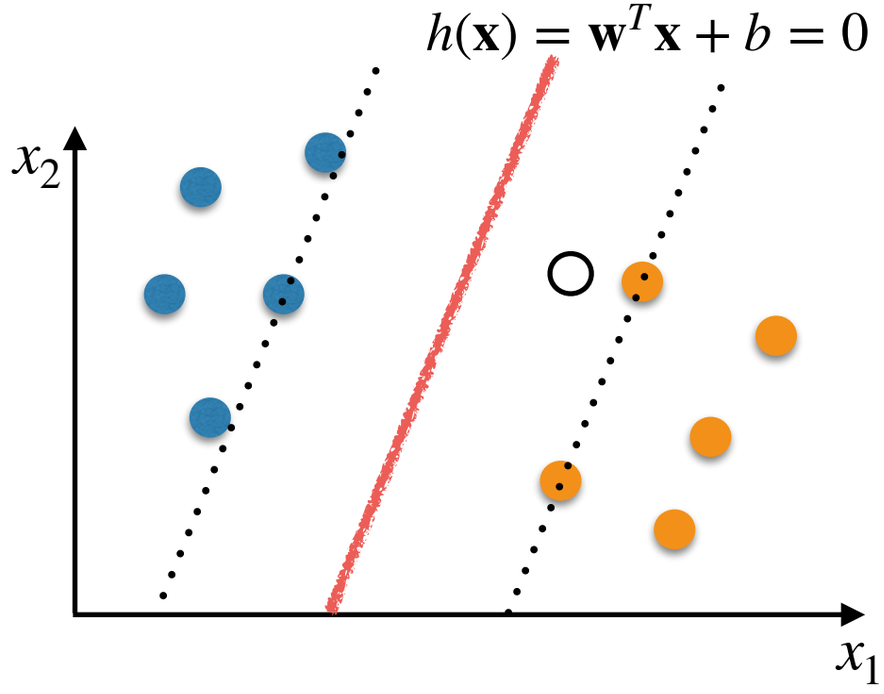
$$b = \frac{1}{N_S} \sum_{n \in S} \left(y^{(n)} - \sum_{m \in S} a^{(m)} y^{(m)} k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) \right)$$

where S is the set of indexes of the support vectors and N_S is the number of support vectors.

Kernels as Similarity Functions

Dual: $h(\mathbf{x}) = \sum_{n \in S} a^{(n)} y^{(n)} k(\mathbf{x}, \mathbf{x}^{(n)}) + b$

- $h(\mathbf{x}) > 0 \rightarrow$ class +1
- $h(\mathbf{x}) < 0 \rightarrow$ class -1



We are now making predictions on new examples based on the training examples.

Is It Possible to Construct Kernel Functions By Designing Similarity Functions Directly?

- … rather than designing their basis expansions ϕ explicitly?
- Even though we don't need to compute ϕ, the kernel function must correspond to **some** embedding.
- In particular, it needs to correspond to the inner product in some embedding.

$$k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) = \phi(\mathbf{x}^{(n)})^T \phi(\mathbf{x}^{(m)})$$

- We can check if it does based on the Mercer's condition.

Mercer's Condition

- Consider any finite set of points $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(M)}$ (not necessarily the training set).
- Gram matrix: An $M \times M$ similarity matrix $\mathbf{K}$, whose elements are given by $K_{i,j} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$.
- Mercer's condition states that $\mathbf{K}$ must be symmetric and positive semidefinite.
  - Symmetric: $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = k(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})$.
  - Positive semidefinite: $\mathbf{z}^T \mathbf{K} \mathbf{z} \geq 0, \forall \mathbf{z} \in \mathbb{R}^M$.

If these conditions are satisfied, the inner product defined by the kernel in the feature space respects the properties of inner products.

Kernel Composition Rules

Given valid kernels $k_1(\mathbf{x}, \mathbf{z})$ and $k_2(\mathbf{x}, \mathbf{z})$, the following will also be valid kernels:

$$k(\mathbf{x}, \mathbf{z}) = ck_1(\mathbf{x}, \mathbf{z})$$ where $c \geq 0$ is a constant.

$$k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{z})f(\mathbf{z})$$ where $f(\cdot)$ is any function.

$$k(\mathbf{x}, \mathbf{z}) = q(k_1(\mathbf{x}, \mathbf{z}))$$ where $q(\cdot)$ is a polynomial with non-negative coefficients.

$$k(\mathbf{x}, \mathbf{z}) = e^{k_1(\mathbf{x}, \mathbf{z})}$$

$$k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$$

$$k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + c$$ where $c \geq 0$ is a constant.

$$k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z})k_2(\mathbf{x}, \mathbf{z})$$

Gaussian Kernel

- Gaussian kernel, a.k.a. Radial Basis Function (RBF) kernel.

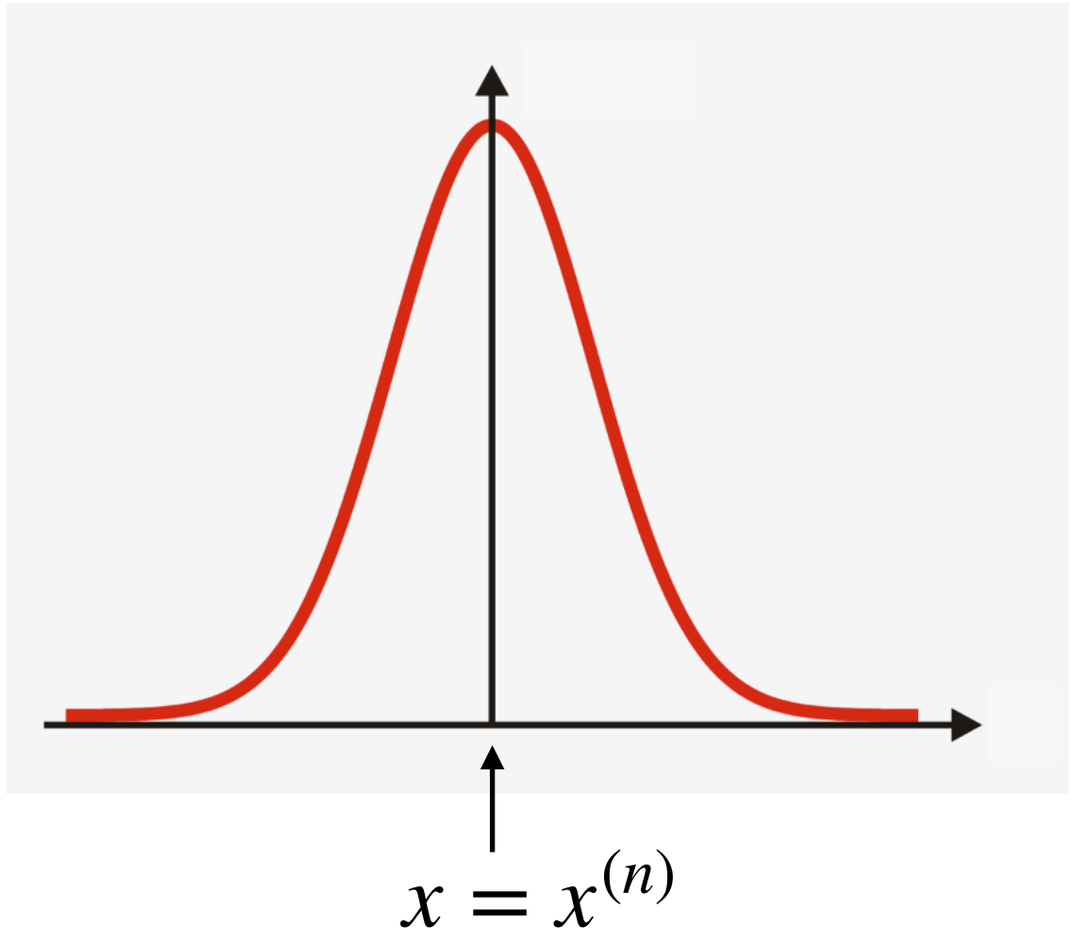
$$k(\mathbf{x}, \mathbf{x}^{(n)}) = e^{-\frac{\|\mathbf{x} - \mathbf{x}^{(n)}\|^2}{2\sigma^2}}$$

The embedding ϕ is infinite dimensional!

Taylor series with infinite terms gives a representation of the true Gaussian itself.

E.g., for $\sigma = 1$

$$k(\mathbf{x}, \mathbf{x}^{(n)}) = \sum_{j=0}^{\infty} \frac{(\mathbf{x}^T \mathbf{x}^{(n)})^j}{j!} e^{-\frac{1}{2}\|\mathbf{x}\|^2} e^{-\frac{1}{2}\|\mathbf{x}^{(n)}\|^2}$$



Gaussian is a kind of similarity measure.

Proving That the Gaussian Kernel is a Valid Kernel, Assuming $\mathbf{x}^T\mathbf{z}$ Is A Valid Kernel

$$k(\mathbf{x},\mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}} = \sum_{j=0}^{\infty} \frac{(\mathbf{x}^T\mathbf{z})^j}{j!} e^{-\frac{1}{2}\|\mathbf{x}\|^2} e^{-\frac{1}{2}\|\mathbf{z}\|^2}$$

$$= \sum_{j=0}^{\infty} e^{-\frac{1}{2}\|\mathbf{x}\|^2} \frac{(\mathbf{x}^T\mathbf{z})^j}{j!} e^{-\frac{1}{2}\|\mathbf{z}\|^2} \qquad k(\mathbf{x},\mathbf{z}) = k_1(\mathbf{x},\mathbf{z}) + k_2(\mathbf{x},\mathbf{z})$$

$$e^{-\frac{1}{2}\|\mathbf{x}\|^2} \frac{(\mathbf{x}^T\mathbf{z})^j}{j!} e^{-\frac{1}{2}\|\mathbf{z}\|^2} \qquad k(\mathbf{x},\mathbf{z}) = f(\mathbf{x})k_1(\mathbf{x},\mathbf{z})f(\mathbf{z})$$

Proving That the Gaussian Kernel is a Valid Kernel

$$e^{-\frac{1}{2}\|\mathbf{x}\|^2}\frac{(\mathbf{x}^T\mathbf{z})^j}{j!}e^{-\frac{1}{2}\|\mathbf{z}\|^2}$$

$$k(\mathbf{x},\mathbf{z}) = f(\mathbf{x})k_1(\mathbf{x},\mathbf{z})f(\mathbf{z})$$

$$\frac{(\mathbf{x}^T\mathbf{z})^j}{j!}$$

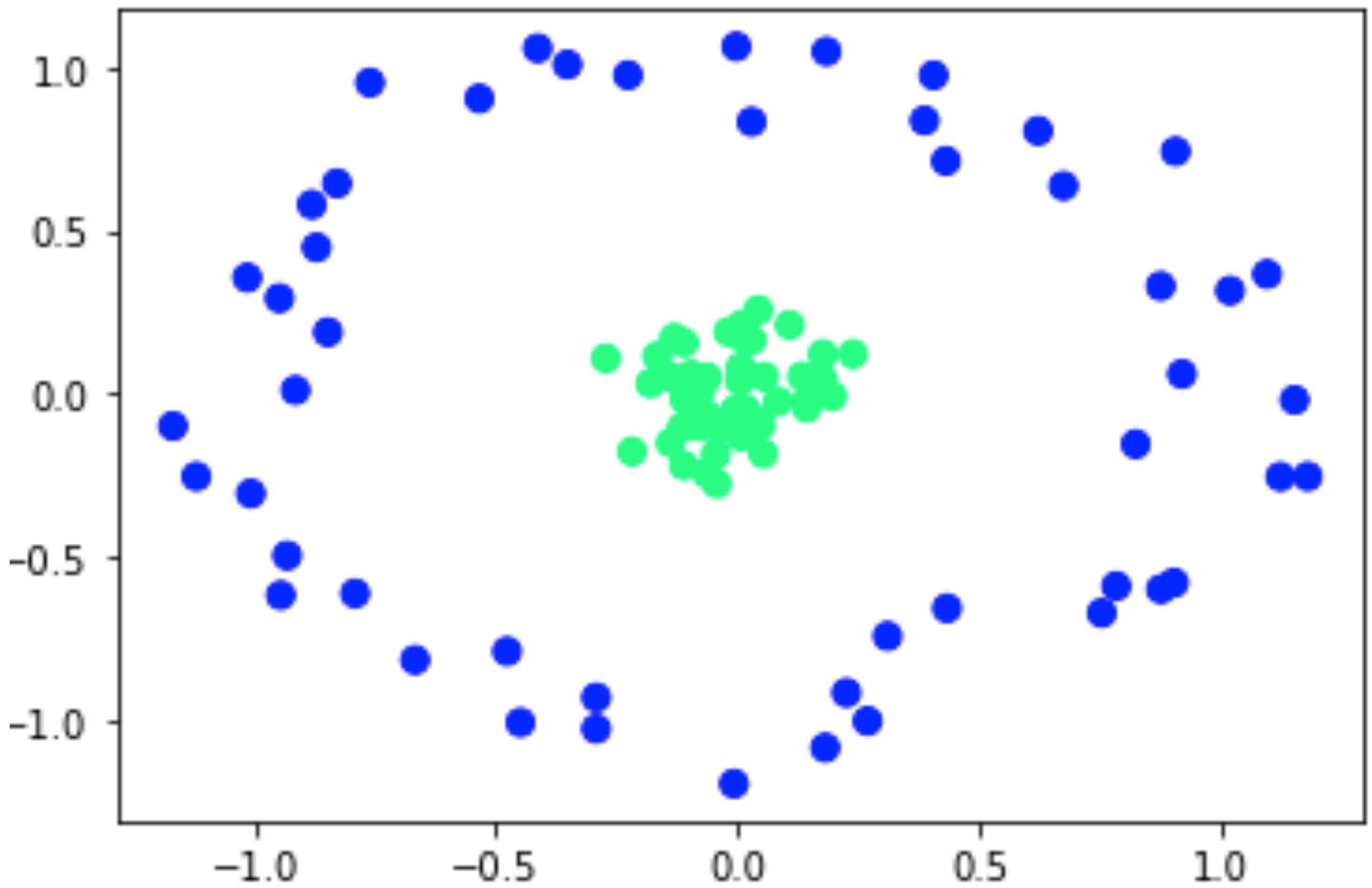
$$k(\mathbf{x},\mathbf{z}) = ck_1(\mathbf{x},\mathbf{z})$$

$$(\mathbf{x}^T\mathbf{z})^j$$

$$k(\mathbf{x},\mathbf{z}) = k_1(\mathbf{x},\mathbf{z})k_2(\mathbf{x},\mathbf{z})$$

$$\mathbf{x}^T\mathbf{z}$$

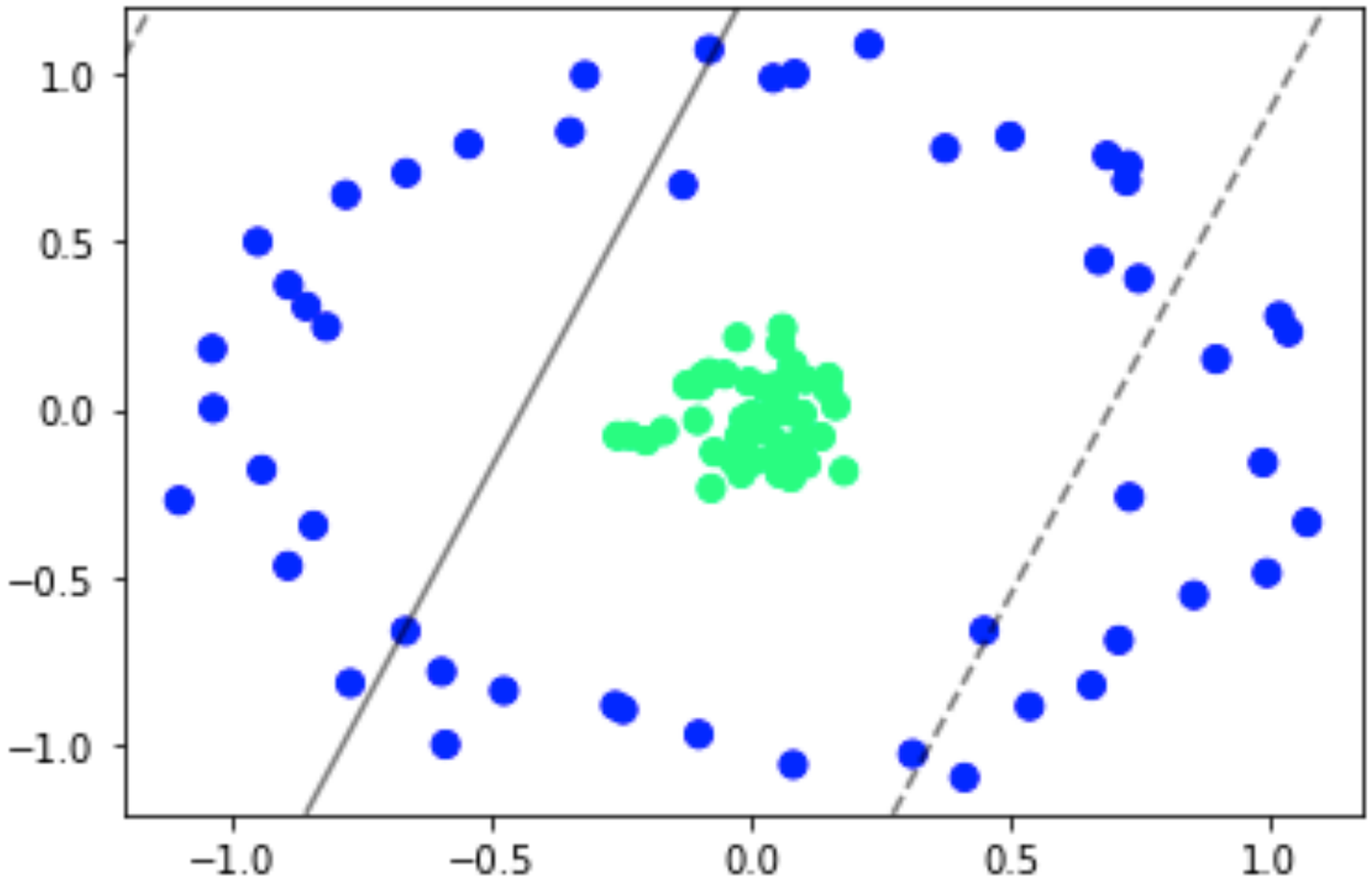
Example



Code adapted from: https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

Example

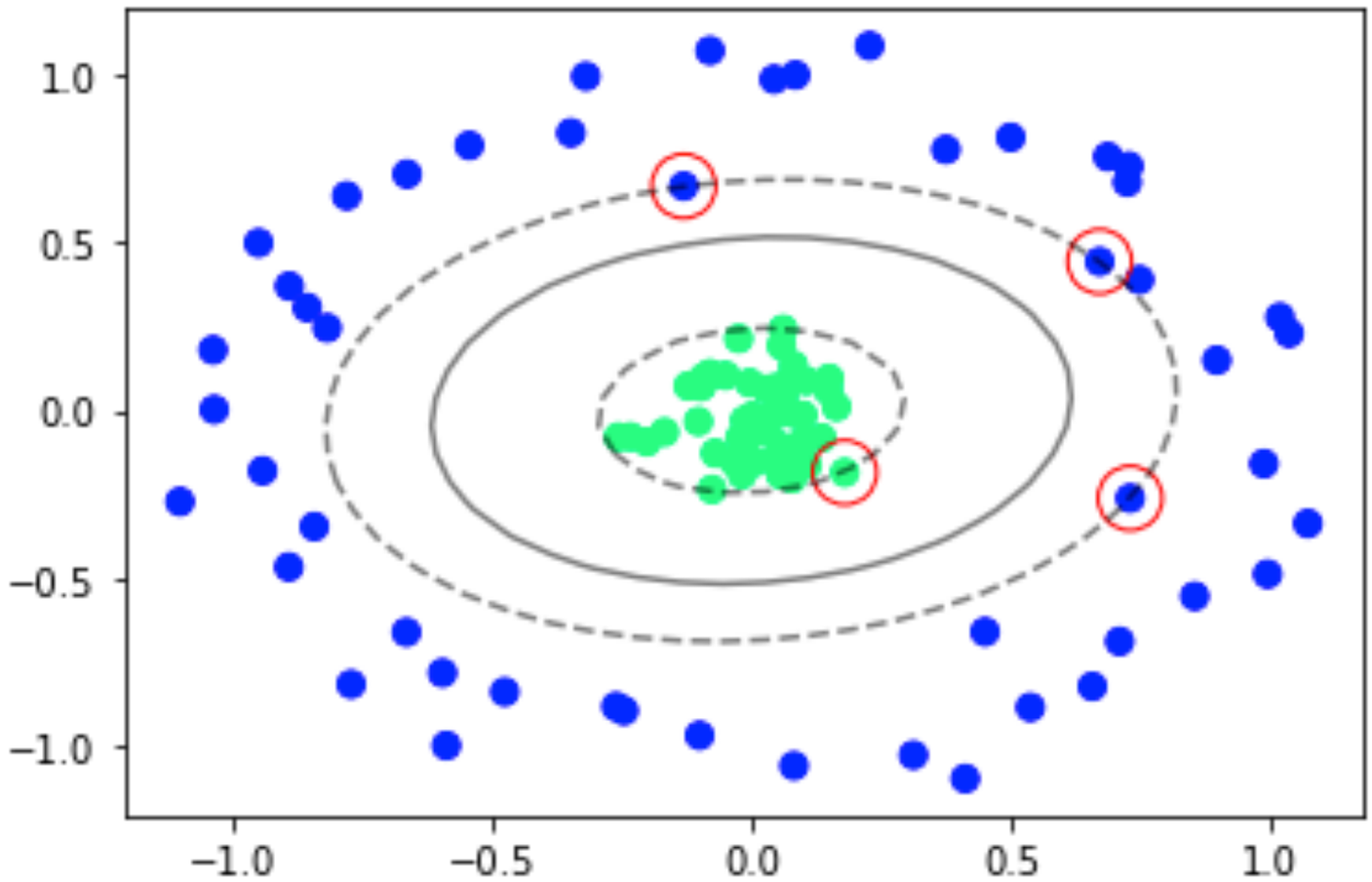
Using $\phi(\mathbf{x}) = \mathbf{x}$, linear kernel: $k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) = \mathbf{x}^{(n)^T}\mathbf{x}^{(m)}$



Code adapted from: https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

Example

Using polynomial embedding of degree 2, $(1 + \mathbf{x}^T\mathbf{z})^2$

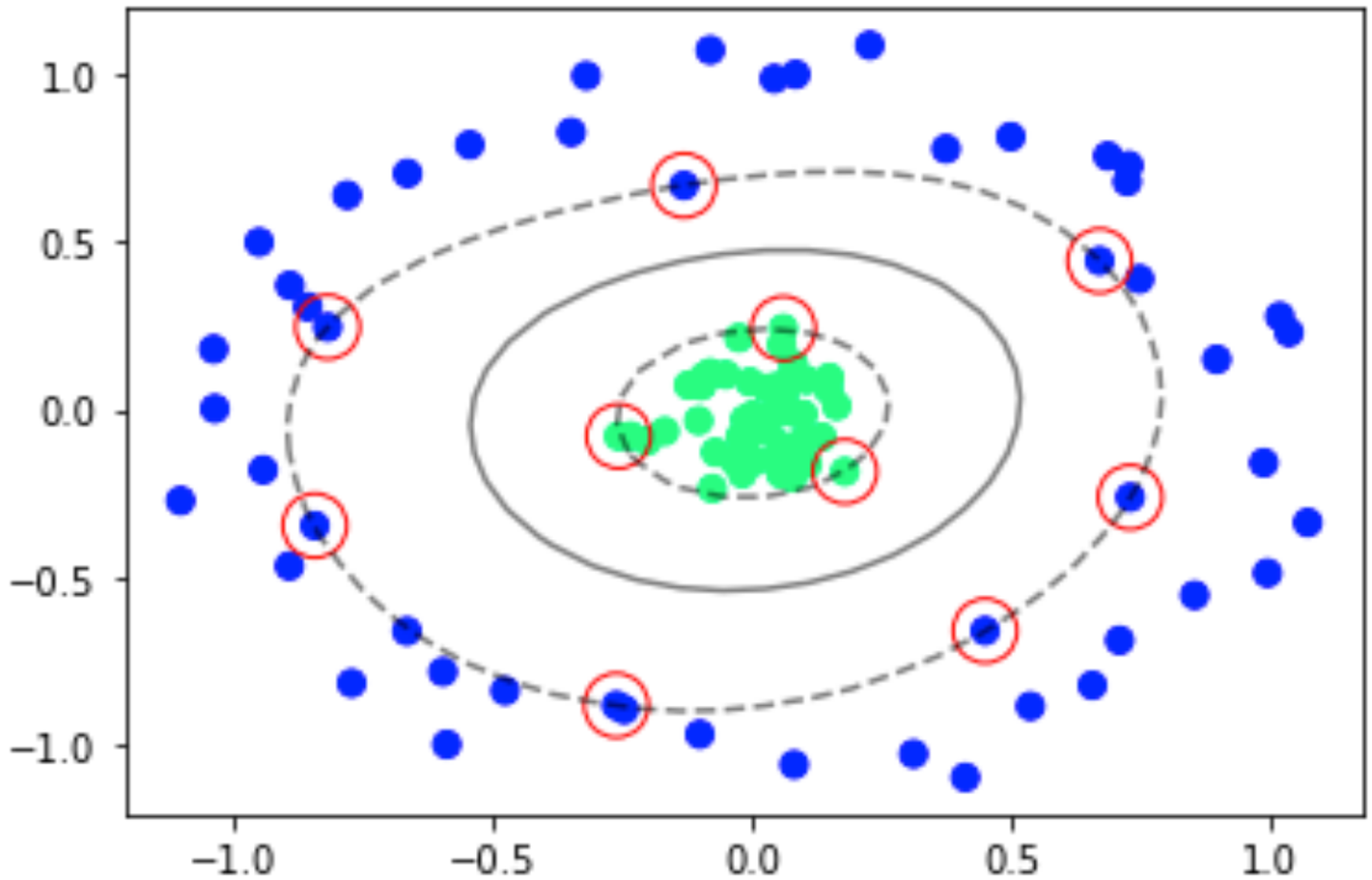


Code adapted from: https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

* Red circles represent the support vectors

Example

Using Gaussian kernel: $k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) = e^{-\frac{\|\mathbf{x}^{(n)} - \mathbf{x}^{(m)}\|^2}{2\sigma^2}}$

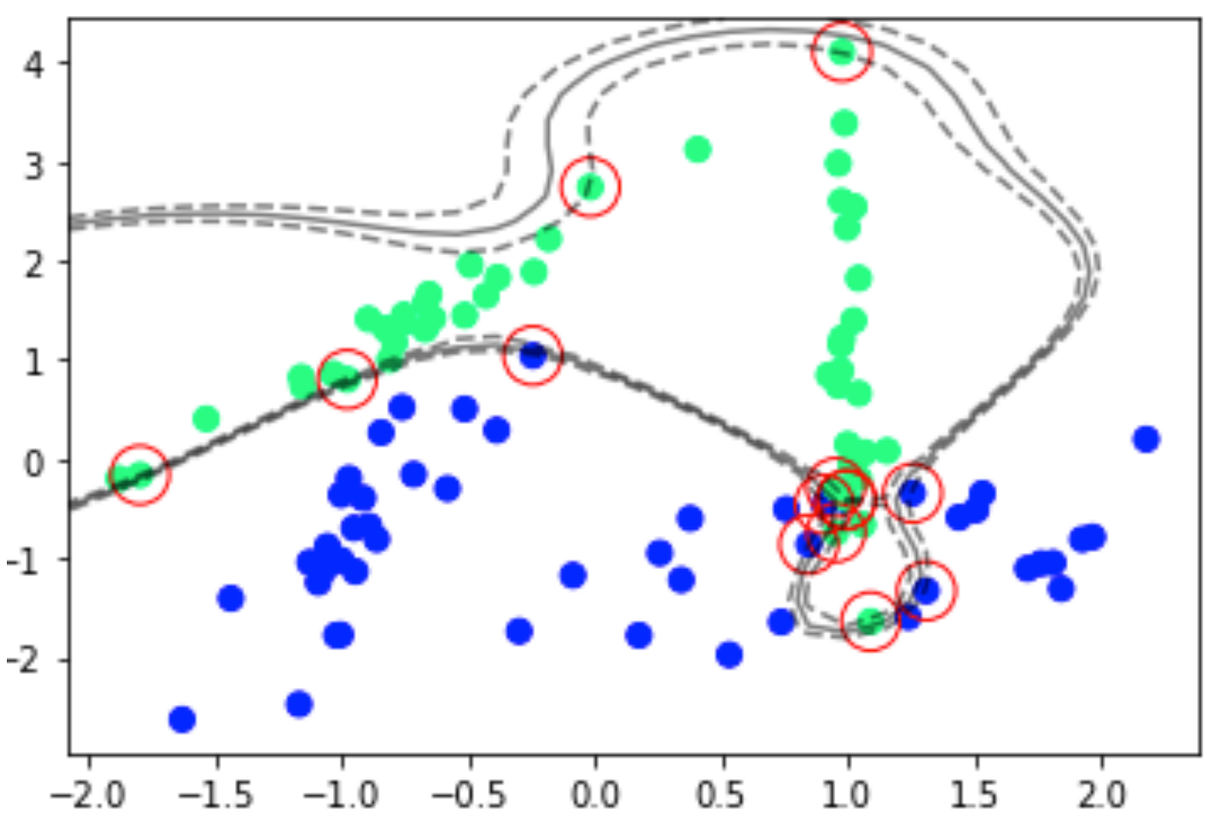


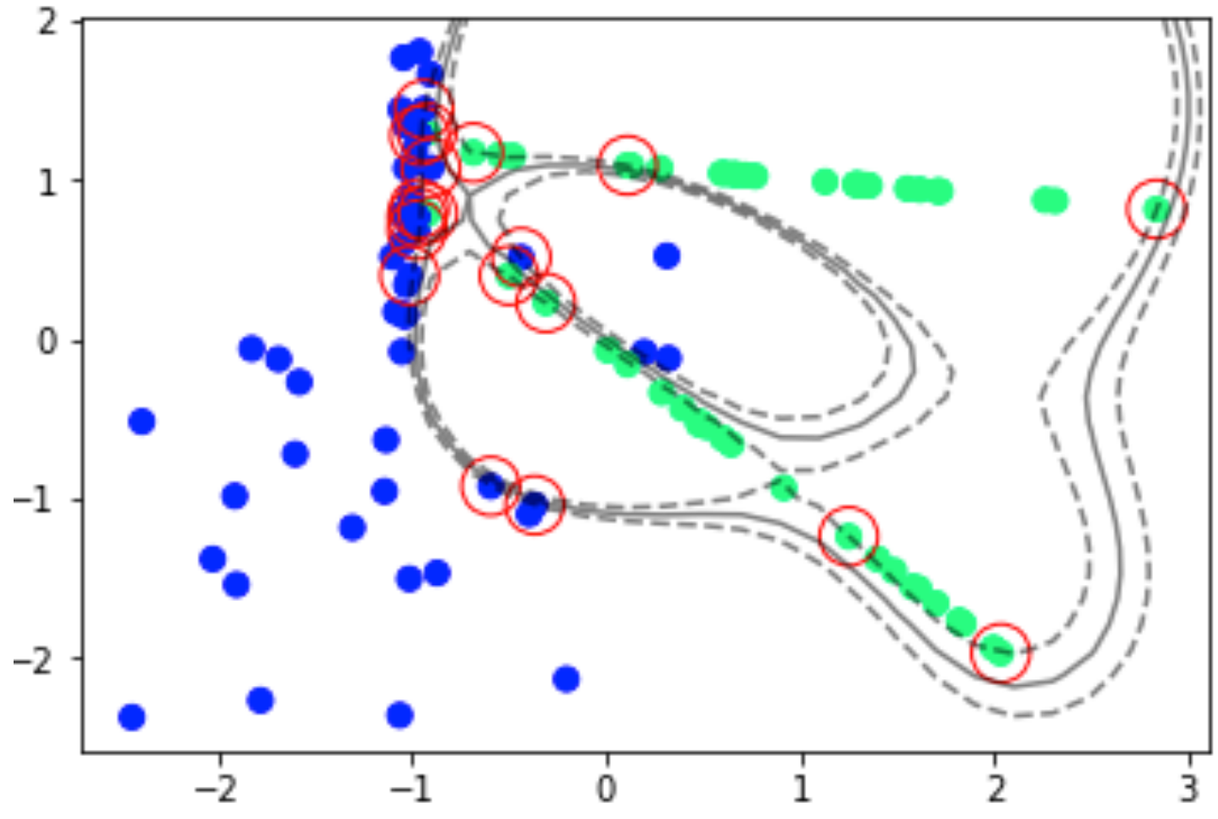
Code adapted from: https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

* Red circles represent the support vectors

Example

Using Gaussian kernel: $k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) = e^{-\frac{\|\mathbf{x}^{(n)} - \mathbf{x}^{(m)}\|^2}{2\sigma^2}}$

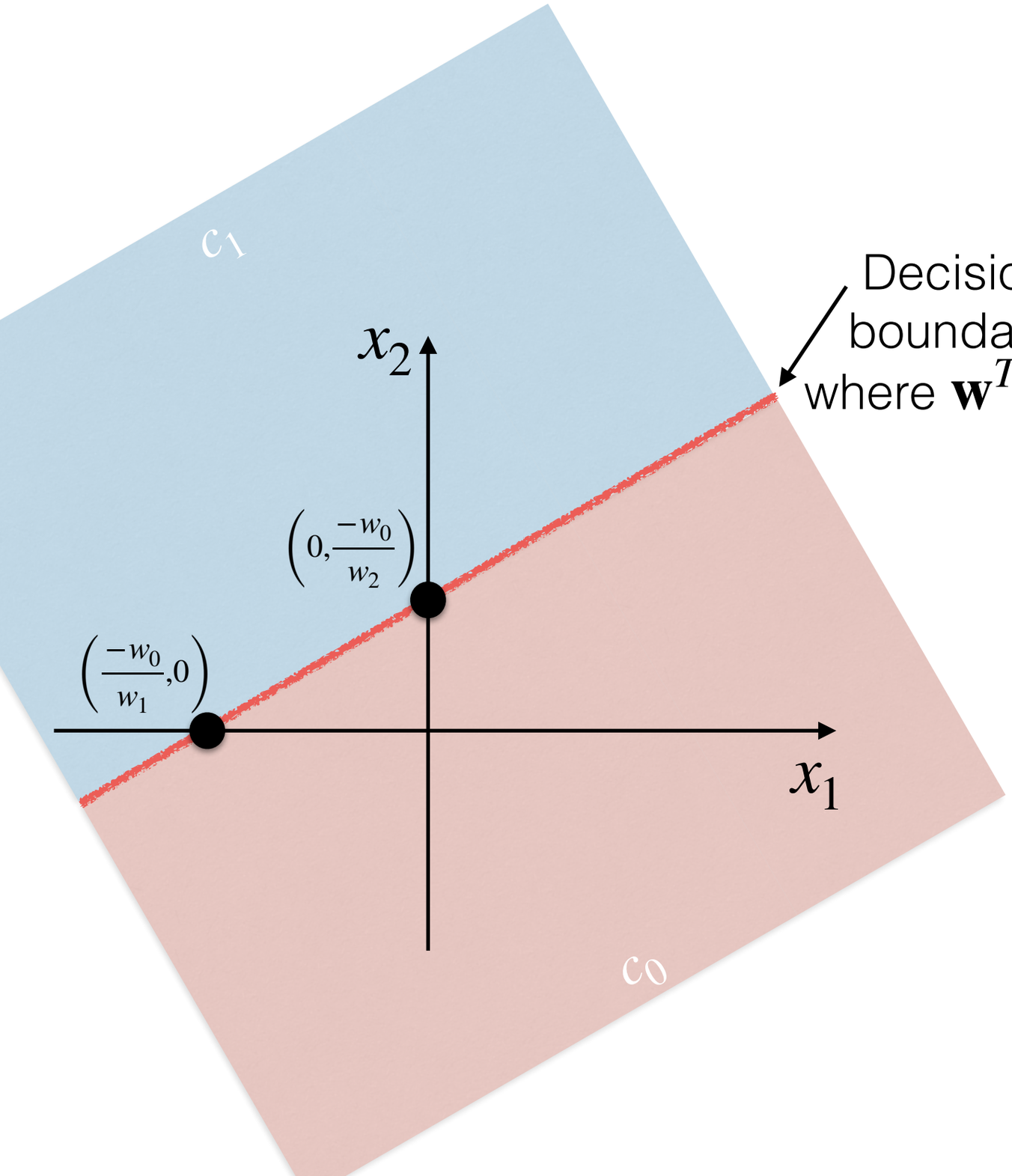


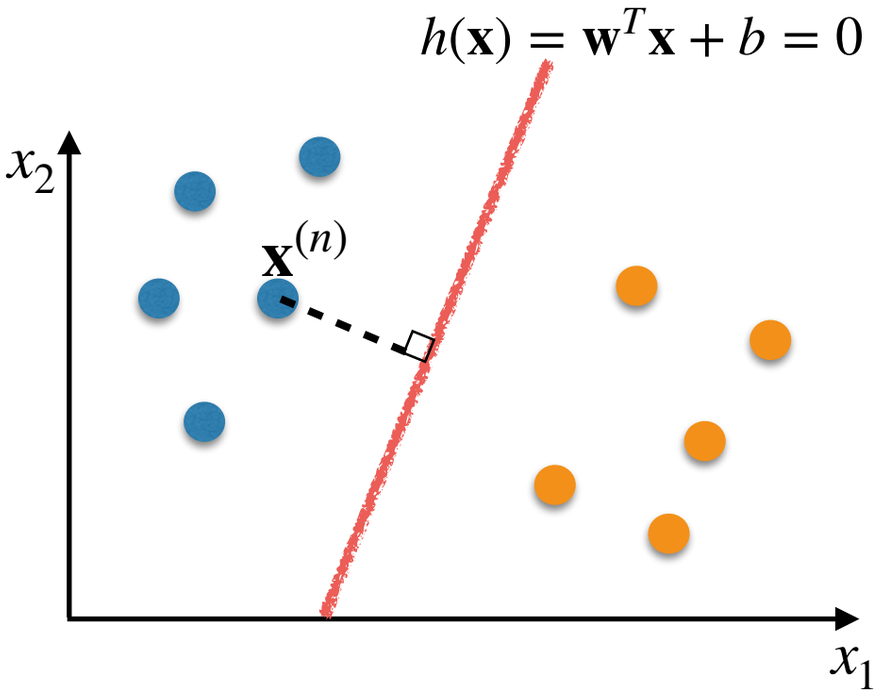


Code adapted from: https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

* Red circles represent the support vectors

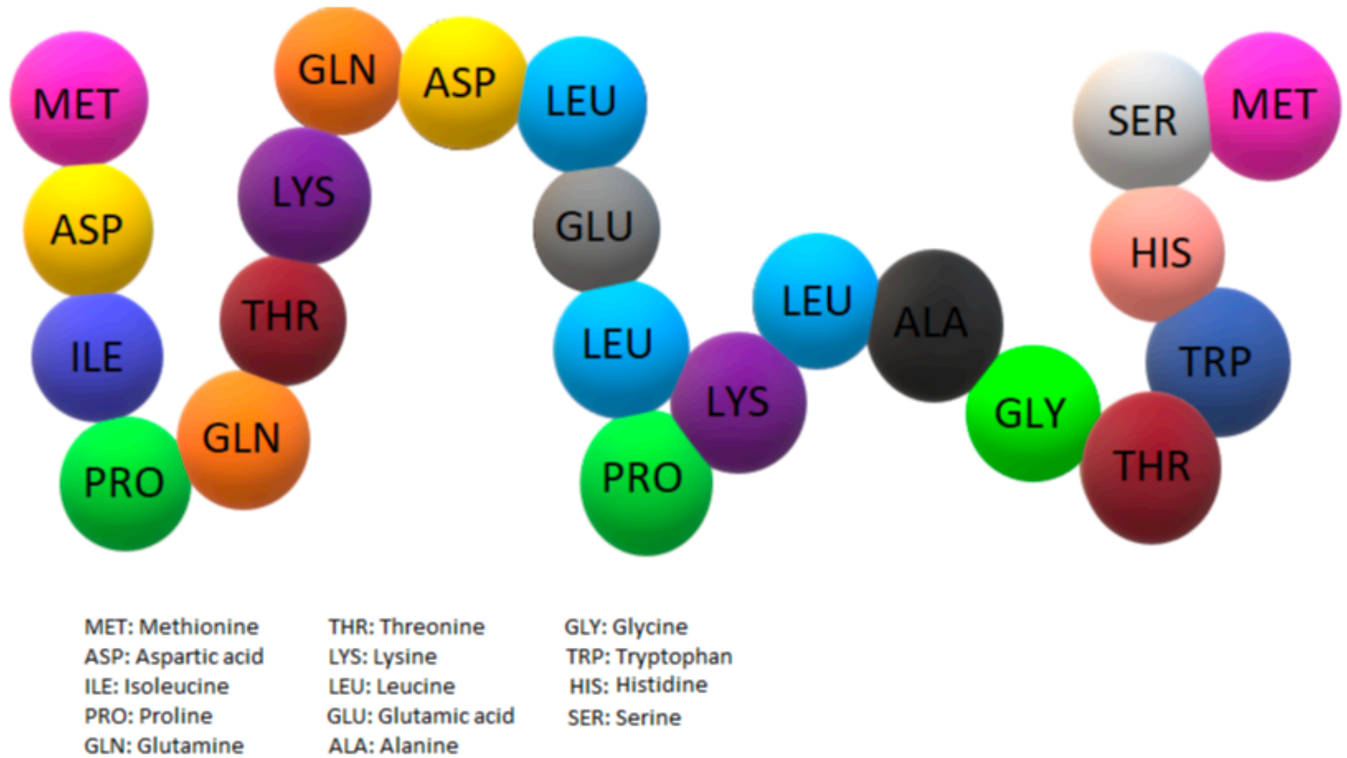
Numeric Inputs





Strings of Various Lengths

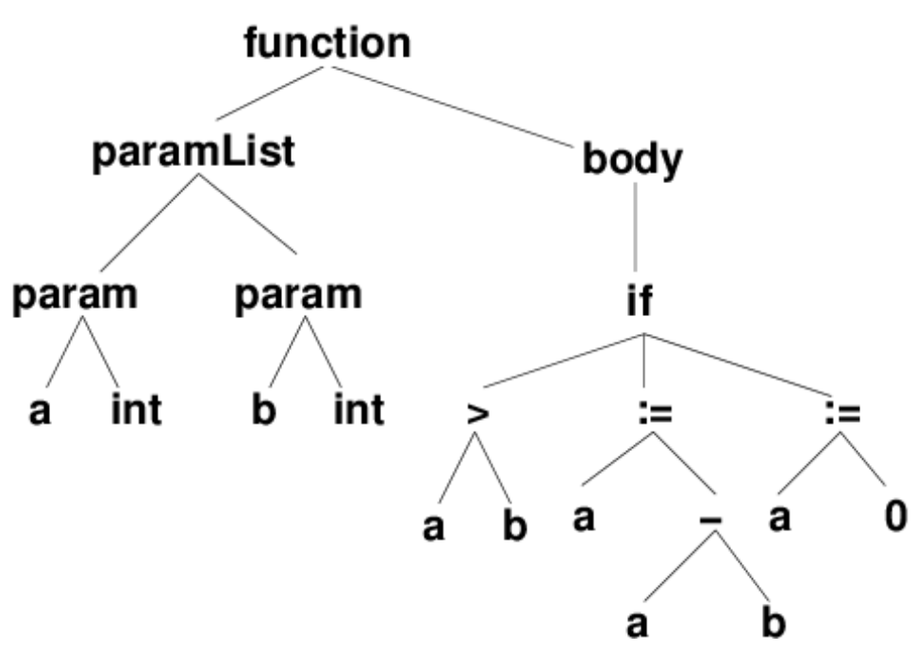
E.g.: classify protein's amino acid sequences into protein family



{ABCDEFGEBHIHDGHJDFKLMA}

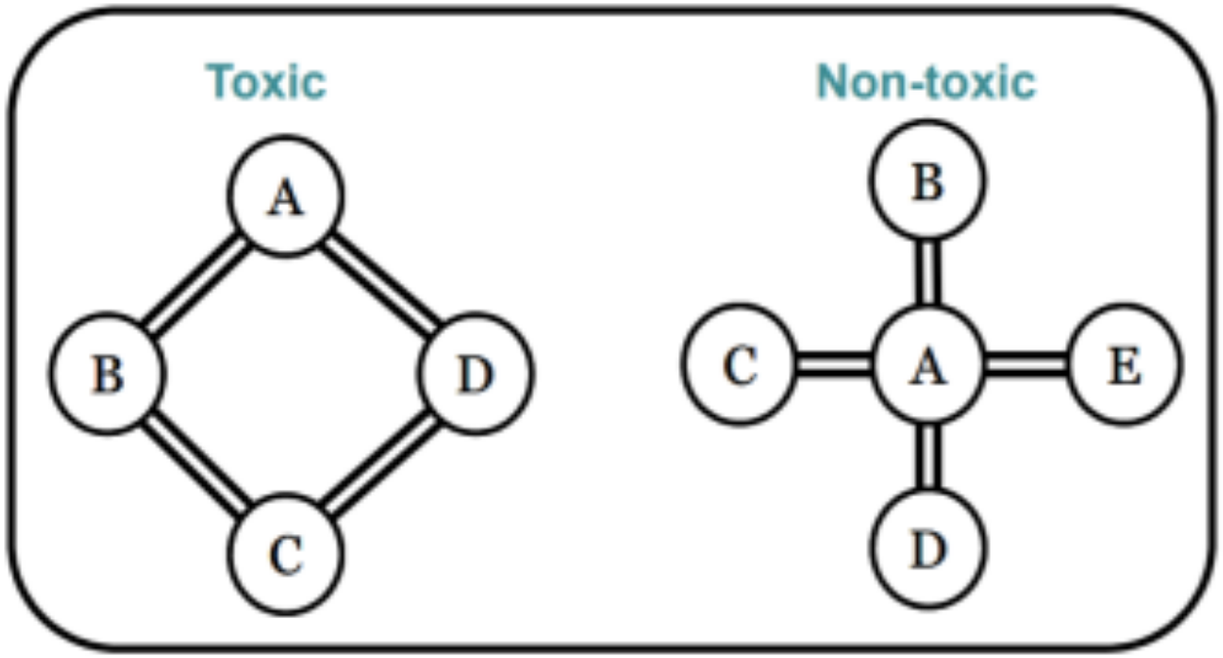
Trees of Various Sizes

E.g.: classify computer program as defective



Graphs of Various Sizes

E.g.: classify chemical compounds into toxic or non-toxic



Source: N. Samatova. “Graph classification.” 2015. Available: https://www.csc2.ncsu.edu/faculty/nfsamato/practical-graph-mining-withR/slides/pdf/Classification.pdf

Choosing Features Embeddings or Kernels For a Given Data Type

- Devise an appropriate feature embedding and compute its kernel.
  - Feature mapping can map data types to numeric spaces.
  - The kernel should ideally be a function that can be computed efficiently.
- Devise an appropriate similarity function that can be computed efficiently for the data type we are interested in and use it as a kernel.
  - It needs to correspond to an inner product in some embedding (Mercer's Condition).
  - Kernels can be defined for inputs that are not in vectorial format, e.g., strings, graphs, text documents and sets.
  - Despite representing inner products in some feature space, we don't necessarily need to reflect about what exactly the feature embedding itself means.

Kernel for Sets

Let A_1 and A_2 be two subsets of a larger set A

$$k(A_1, A_2) = 2^{|A_1 \cap A_2|}$$

where $|A_i|$ denotes the number of elements in A_i.

NGED TO INCLUDE THE EMPTY STRING IN A* AND IN THE EXAMPLE BELOW

Strings and A Possible Embedding

- A string s is a sequence of characters from a given alphabet $\mathscr{A}$.
- Denote the (infinite) set of all possible strings of all possible sizes (including the empty string) as $\mathscr{A}^*$.
- Consider an embedding $\phi(s)$ where each dimension corresponds to a different string s' in $\mathscr{A}^*$.
  - Each dimension contains the number of times that s' appears in s.
  - E.g.: $s = ADA$, $\mathscr{A} = \{A, B, C, D\}$.

$$\phi(s) = \underset{\{\}}{1} \; \underset{A}{2} \; \underset{B}{0} \; \underset{C}{0} \; \underset{D}{1} \; \underset{AA}{1} \; \underset{AB}{0} \; \ldots$$

All-Subsequence Kernel for Strings

The all-subsequence kernel for strings is:

$$k(s,t) = \phi(s)^T \phi(t) = \sum_{i=1}^{\infty} \phi_i(s)\phi_i(t)$$

We may speed up calculation if we consider only the terms that make use of characters that appear in s and t and only substrings up to size $\min(|s|,|t|)$.

Still, the time complexity would be likely at least exponential on $\min(|s|,|t|)$.

All-Subsequence Kernel for Strings

- Dynamic programming enables us to compute this more efficiently in $O(|s||t|)$.
  - A technique where the solution to a bigger problem can be computed based on the solution for a smaller problem.
  - This enables us to compute for larger subsequences based on what we have already computed for smaller subsequences.

ABCDEF AB ABC

All-Subtree Kernel

Assume T_1 and T_2 are trees that can be constructed with a given set of possible nodes, and $\mathcal{T}$ is the set of all possible trees.

$$\phi_S(T) = 1(S \in T) \qquad (S \text{ is a subtree of } T)$$

$$k(T_1, T_2) = \phi(T_1)^T \phi(T_2) = \sum_{S \in \mathcal{T}} \phi_S(T_1)\phi_S(T_2)$$

This kernel is a similarity metric that counts the number of sub-trees in common between T_1 and T_2.

All-Subtree Kernel

Assume T_1 and T_2 are trees that can be constructed with a given set of possible nodes, and $\mathcal{T}$ is the set of all possible trees.

$$\phi_S(T) = 1(S \in \mathcal{T}) \qquad (S \text{ is a subtree of } \mathcal{T})$$

$$k(T_1, T_2) = \phi(T_1)^T \phi(T_2) = \sum_{S \in \mathcal{T}} \phi_S(T_1)\phi_S(T_2)$$

Dynamic programming can be used to compute this kernel in $O(|T_1||T_2|d_{max}^2)$, where d_{max} is the maximum number of children a node has in these trees.

Kernels for Graphs and Other Structures

Kernels for graphs and other structures that can be decomposed into smaller sub-structures can be defined using similar ideas to the ones described for kernels for trees.

Summary

- Predictions when using the dual representation are based on the support vectors.

- The dual representation enables us to adopt powerful kernels, e.g., Gaussian kernel, which takes us to an infinite dimensional embedding.

- Kernels are powerful tools:
  - Can be seen as similarity measures.
  - Can potentially enable us to compute inner products efficiently.
  - Can enable us to deal with different data types.

Different Terminologies

- The term kernel can also be used with other meanings.
- Some simply define a kernel as a real valued function of two arguments $k(\mathbf{x}, \mathbf{z}) \in \mathbb{R}$.
- Some simply define a kernel as a real valued function of two arguments $k(\mathbf{x}, \mathbf{z}) \geq 0$.
- The term kernel is also used to refer to a matrix which is slid across an image and multiplied with the input such that the output is enhanced in a desired manner (used in Convolutional Neural Networks).