(Generalized) Linear Models and SVM

Machine Learning, A.Y. 2022/23, Padova

Fabio Aiolli

9th / 14th November, 2022
Outline

- Linear methods for classification and regression
- Non-linear transformations
- Optimal hyperplane and Support Vector Machine (SVM)
- SVM for non linearly-separable data
Linear Models (recap)

- One of the most important types of models in ML
- A linear model is in the form $f_{w,b}(x) = \sum_{i=1}^{m} w_i x_i + b = w \cdot x + b$
- Which can also be written as $f_w(x) = \sum_{i=0}^{m} w_i x_i = w \cdot x$ where $x_0 = 1$ is an ad-hoc artificial feature (coordinate)
- For classification, the sign is returned, that is $h(x) = \text{sign}(f_w(x)) \in \{-1, +1\}$
- For regression, the original function can be taken, that is $h(x) = f_w(x) \in \mathbb{R}$
The Perceptron Algorithm

\[ h_w(x) = \text{sign}(\sum_{i=0}^{n} w_i x_i) = \text{sign}(w \cdot x) \]

Given a training set \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

1. Let \( w = 0 \)
2. Pick a misclassified point \( x_i \) (\( \text{sign}(w \cdot x_i) \neq y_i \))
3. Update the weight vector \( w \leftarrow w + y_i x_i \)
4. Repeat from step 2 until all points are correctly classified

If data are linearly separable, then the perceptron algorithm always converges to a valid solution!
Linear models for classification

- Perceptron
- Constrained linear problem
- Least squares for classification
- Logistic regression models
- SVM
- ...

Fabio Aiolli  (Generalized) Linear Models and SVM  9th / 14th November, 2022
(Multi-variate) Linear Regression

- Example: Given a set of characteristics for a user: age, annual salary, years in residence, years in job, current debits, etc. Predict the credit line, that is the amount of credit that can be granted to a customer.
- Given \( \text{TRAIN} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \), in linear regression we look for a hypothesis \( h_w \) (a linear space) which minimizes the mean squared error on the training set, that is:

\[
\arg \min_w \frac{1}{n} \sum_{i=1}^{n} (h_w(x_i) - y_i)^2
\]
Solving the Linear Regression problem

\[ E(w) = \frac{1}{n} \sum_{i=1}^{n} (h_w(x_i) - y_i)^2 \]

\[ = \frac{1}{n} \sum_{i=1}^{n} (w \cdot x_i - y_i)^2 \]

\[ = \frac{1}{n} \|Xw - y\|^2 \]

where

\[ X = \begin{pmatrix} \vdots & \x_1 \top & \cdots \\ \x_2 \top & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \x_n \top & \cdots & \vdots \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \]
By minimizing the residual error

$$\min_w E(w) \equiv \frac{1}{n} \|Xw - y\|^2$$

$$\nabla E(w) = \frac{2}{n} X^\top (Xw - y) = 0$$

$$X^\top Xw = X^\top y$$

$$w = X'y, \text{ where } X' = (X^\top X)^{-1}X^\top$$

The matrix $X'$ is the pseudo-inverse of $X$
Real Data

- $16 \times 16$ grey-level images
- Standard representation: raw input $\mathbf{x} = (x_1, \ldots, x_{256})$
- You can also use other representations (e.g. intensity, symmetry)
Non-linear Mapping

\[(x_1, x_2) \xrightarrow{\Phi} (x_1^2, x_2^2)\]

Generalized Linear Models: In general, any non-linear transformation \(\mathbf{x} \xrightarrow{\Phi} \mathbf{z}\) (a.k.a. basis function) can be applied to the data. An hyperplane, hence a linear model, in the transformed space will correspond to a non-linear decision surface in the original space!
Consider the hypothesis space of hyperplanes

Take a set of linearly separable points

We have different separating hyperplanes fitting the data

Which is the best?

Two questions:

1. The widest possible margin (or optimal) hyperplane is the best, why?
2. To which $\mathbf{w}, b$ this corresponds?
Given the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$, the “distance” of a point $\mathbf{x}$ from the hyperplane can be expressed by the algebraic measure $g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$.

We can write $\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{||\mathbf{w}||}$ where:

- $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto the hyperplane
- $r$ is the desired algebraic distance ($r > 0$ if $\mathbf{x}$ is on the positive side of the hyperplane, otherwise $r < 0$)
Margin

\[ g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \quad \text{and} \quad \mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{||\mathbf{w}||}, \]

Two facts:

- Note that \( g(\mathbf{x}_p) = 0 \) (because \( \mathbf{x}_p \) is on the optimal hyperplane)
- Since the absolute distance from any nearest positive example is the same as the absolute distance from any nearest negative example, then we can consider hypotheses \( \mathbf{w}, b \) such that \( g(\mathbf{x}) = 1 \) when \( \mathbf{x} \) is in the (positive side) margin hyperplane and \( g(\mathbf{x}) = -1 \) when \( \mathbf{x} \) is in the (negative side) margin hyperplane.

Take \( \mathbf{x}_k \) in the positive margin hyperplane, then

\[ g(\mathbf{x}_k) = \mathbf{w} \cdot \mathbf{x}_p + b + r \frac{\mathbf{w} \cdot \mathbf{w}}{||\mathbf{w}||} = r ||\mathbf{w}|| \quad \Rightarrow \quad r = \frac{g(\mathbf{x}_k)}{||\mathbf{w}||} = \frac{1}{||\mathbf{w}||} \]

and hence the margin will be \( \rho = 2r = \frac{2}{||\mathbf{w}||} \).
Can we apply the **Structural Risk Minimization (SRM)** principle to hyperplanes?

We have seen that the hypothesis space of hyperplanes in $\mathbb{R}^m$ has $VC = m + 1$.

In fact, if we add further constraints on the hyperplanes we can do better!
Theorem: Let $R$ denote the diameter of the smallest ball containing all the input points. The set of hyperplanes described by the equation $w \cdot x + b = 0$ with margin at least $\rho$ has a VC-dimension $VC_\rho$ bounded from above as

$$VC_\rho \leq \min\{ \lceil \frac{R^2}{\rho^2} \rceil, m \} + 1$$

where $m$ is the dimensionality of the input space.

Thus, if we consider the hypothesis spaces

$$\mathcal{H}_k = \{ w \cdot x + b \mid \|w\|^2 \leq c_k \}$$

where $c_1 < c_2 < c_3 < \ldots$

and linearly separable data, then the empirical error of $\mathcal{H}_k$ is 0 for each $k$ and the bound on the true risk can be minimized by maximizing the margin of separation (i.e. minimizing the weight norm).
Separable Case: Quadratic optimization

If we have \( n \) linearly separable examples \( \{(x_i, y_i)\}_{i=1}^{n} \), it is possible to find the optimal hyperplane solving the following constrained quadratic optimization problem:

\[
\min_{w, b} \frac{1}{2} ||w||^2 \\
\text{subject to: } \forall i \in \{1, \ldots, n\} : y_i (w \cdot x_i + b) \geq 1
\]

- This is a \textbf{(convex) constrained quadratic problem}. This guarantees a \textbf{unique} solution!
- Many QP algorithms exist with polynomial complexity to find the solution of this quadratic problem
Solving the optimization problem

- The problem above, called **primal problem**, can be solved more easily using the **dual formulation**.
- In the dual problem, Lagrange multipliers $\alpha_i \geq 0$ are associated with every constraint in the primal problem (one for each example).

The dual formulation is:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

subject to: $\forall i \in \{1, \ldots, n\}: \alpha_i \geq 0$ e $\sum_{i=1}^{n} y_i \alpha_i = 0$.

- At the solution, most of the $\alpha_i$’s are zeros. Those examples associated with non zero multipliers are called **support vectors**.

Fabio Aiolli
(Generalized) Linear Models and SVM
9th / 14th November, 2022 17 / 50
The primal solution turns out to be:

\[ w = \sum_{i=1}^{n} y_i \alpha_i x_i \]
\[ b = y_k - w \cdot x_k \quad \text{for any } x_k \text{ such that } \alpha_k > 0 \]

and hence:

\[ h(x) = \text{sign}(w \cdot x + b) = \text{sign}(\sum_{i=1}^{n} y_i \alpha_i (x_i \cdot x) + b) \]

that is, the decision function only depends on dot products between the point and other points in the training set (the support vectors).
SVM for the Non-separable case

If the examples are NOT linearly separable we have to allow that some constraints are violated. This can be done by

- introducing *slack* variables $\xi_i \geq 0$, $i = 1, \ldots, n$, one for each constraint:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i$$

- modifying the cost function so to penalize slack variables which are not 0:

$$\frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \xi_i$$

where $C$ (regularization parameter) is a positive constant controlling the tradeoff between the complexity of the hypothesis space and the number of margin errors.
SVM for the Non-separable Case

The dual of this new formulation is very similar to the previous one:

$$\max_\alpha \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

subject to: \(\forall i \in \{1, \ldots, n\}: 0 \leq \alpha_i \leq C \) e \(\sum_{i=1}^n y_i \alpha_i = 0\).

The main difference is due to the fact that the dual variables are upper bounded by \(C\). The value for \(b\) is obtained similarly to the separable case (with some minor differences...).
Analysis of SVM for the non-separable case

- The parameter C can be seen as a way to control overfitting.
- As C becomes larger it is unattractive to not respect the data at the cost of reducing the geometric margin.
- When C is small, larger margin is possible at the cost of increasing errors in training data.
- Interestingly, the SVM solution is in the same form as in the hard margin case!

Nevertheless, this formulation is not always satisfactory because of the limited separation capability of a hyperplane.
Another approach

How can we separate these data?
Another approach

Projecting them into a higher dimensional space

\[ \mathbf{x} \rightarrow \varphi(\mathbf{x}) \]
Nonseparable case

When the examples are not linearly separable, an alternative approach based on the following two steps can be used:

1. the input vectors (input space) are projected onto a larger space (feature space);
2. the optimal hyperplane in feature space is computed (e.g. using the formulation with slack variables)
Nonseparable case

Step 1 is justified by Cover’s theorem on separability, which states that non-linearly separable patterns may be transformed into a new feature space where the patterns are linearly separable with high probability, provided that the transformation is nonlinear, and that the dimensionality of the feature space is high enough.

Step 2 is justified by the fact that the optimal hyperplane (in the feature space) minimizes the VC-dimension.
Nonseparable case

We can assume that any of the new feature space coordinate is generated by a nonlinear function $\varphi_j(\cdot)$. Thus, we can consider $M$ functions $\varphi_j(x)$ with $j = 1, \ldots, M$. A generic vector $x$ is thus mapped into the $M$-dimensional vector

$$\varphi(x) = [\varphi_1(x), \ldots, \varphi_M(x)]$$

Step 2 asks to find the optimal hyperplane into the $M$-dimensional feature space. A hyperplane into the feature space is defined as

$$\sum_{j=1}^{M} w_j \varphi_j(x) + b = 0$$

or, equivalently

$$\sum_{j=0}^{M} w_j \varphi_j(x) = w \cdot \varphi(x) = 0$$

where we added a coordinate $\varphi_0(x) = 1$ and $w_0 = b$. 
Nonseparable case

By

$$w = \sum_{k=1}^{n} y_k \alpha_k \varphi(x_k)$$

the equation defining the hyperplane becomes

$$\sum_{k=1}^{n} y_k \alpha_k \varphi(x_k) \cdot \varphi(x) = 0$$

where $\varphi(x_k) \cdot \varphi(x)$ represents the dot product (in feature space) between vectors induced by the $k$-th training instance and the input $x$. 
Now, what we need is a function $K(\cdot, \cdot)$ (called kernel function) such that

\[ K(x_k, x) = \varphi(x_k) \cdot \varphi(x) = \sum_{j=0}^{M} \varphi_j(x_k) \varphi_j(x) = K(x, x_k) \] (symmetric function)

If we get such a function, we could compute the decision function in feature space WITHOUT explicitly representing the vectors into the feature space:

\[ \sum_{k=1}^{n} y_k \alpha_k K(x_k, x) \]

Functions with this property do actually exist, if some conditions are satisfied... namely, Mercer’s conditions.
Kernel functions

In general, a kernel function satisfying Mercer’s conditions represents a dot-product between vectors generated by some (non-linear) transformation.

Note that we do not need to know such a transformation!!

Examples of popular kernel functions:

- Linear kernel, \( K(x, z) = x \cdot z \)
- Polynomial kernel of degree \( s \in \mathbb{N} \), \( K(x, z) = (x \cdot z + c)^s, \quad c > 0 \)
- Exponential kernel, \( K(x, z) = \exp(x \cdot z) \)
- Radial-basis function (RBF) kernel, \( K(x, z) = \exp(-\gamma \|x - z\|^2), \quad \gamma > 0 \)
The introduction of a kernel does not modify the problem formulation:

$$\max_\alpha \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$
subject to: \( \forall i \in \{1, \ldots, n\} : 0 \leq \alpha_i \leq C \) and \( \sum_{i=1}^{n} y_i \alpha_i = 0 \).

where the needed kernel values are computed over all pairs of vectors \((K(x_i, x_j), \text{ with } i, j = 1, \ldots, n)\) and arranged into a matrix \( K \in \mathbb{R}^{n \times n} \) (symmetric and positive definite) known as kernel matrix or gram matrix.
Formulation with Kernel

E.g., if we use a polynomial kernel with degree $p = 3$ we obtain $K_{i,j} = (x_i \cdot x_j + 1)^3$ and a new instance $x$ is classified by the following discriminant function

$$h(x) = \text{sign}( \sum_{x_k \in SV} y_k \alpha^*_k K(x_k, x)) = \text{sign}( \sum_{x_k \in SV} y_k \alpha^*_k (x_k \cdot x + 1)^3)$$

where $SV$ is the set of support vectors and $\alpha^*_k$ are the optimal values for the support vectors (the remaining dual variable are $0 \Rightarrow$ corresponding vectors, and kernels, do not contribute to the sum).
Formulation with Kernel

Using this approach, we can use a nonlinear transformation $\varphi(\cdot)$ IMPLICITLY, in fact what we need is not the explicit representation of vectors in feature space, but their dot product into the feature space. This can be directly computed in the input space via the kernel function.

Examples of decision surfaces generated IN THE INPUT SPACE with or without kernel (polynomial with degree 3) both for the separable and nonseparable case:
Example of mapping: polynomial kernel

Given two vectors \( \mathbf{x} \) and \( \mathbf{z} \) and the following mapping \( \varphi() \)

\[
\mathbf{x} = (x_1, x_2); \quad \varphi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \\
\mathbf{z} = (z_1, z_2); \quad \varphi(\mathbf{z}) = (z_1^2, z_2^2, \sqrt{2}z_1z_2)
\]

A dot product between \( \varphi(\mathbf{x}) \) and \( \varphi(\mathbf{z}) \) corresponds to evaluate the function \( K_2(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2 \)

\[
\langle \varphi(\mathbf{x}), \varphi(\mathbf{z}) \rangle = \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \rangle = \\
= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1z_1x_2z_2 = (x_1z_1 + x_2z_2)^2 = \langle \mathbf{x}, \mathbf{z} \rangle^2 = K_2(\mathbf{x}, \mathbf{z})
\]

\( K_2() \) is faster to evaluate than \( \langle \varphi(\mathbf{x}), \varphi(\mathbf{z}) \rangle \)!
Examples of $\varphi()$: Polynomial and exponential Kernel

Homogeneous polynomial kernels: $k(x, z) = \langle x, z \rangle^s$ can be constructed by defining an embedding map, indexed by all monomials of degree $s$:

$$\varphi_i(x) = \beta_i \prod_{k=1}^{n} x_k^{i_k}$$

such that $i = (i_1, \ldots, i_n)$ and $\sum_{k=1}^{n} i_k = s$, and opportune $\beta$’s

Non-homogeneous polynomial kernels: $k(x, z) = (\langle x, z \rangle + c)^s$ can be constructed by defining an embedding map, indexed by all monomials of degree less or equal to $s$:

$$\varphi_i(x) = \beta_i \prod_{k=1}^{n} x_k^{i_k}$$

such that $i = (i_1, \ldots, i_n)$ and $\sum_{k=1}^{n} i_k \leq s$, and opportune $\beta$’s

Exponential and RBF kernels: similar embedding to the ones given above but with all possible monomials/degrees (infinite number of features)!
We are given a set of objects $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$. How can they be represented?

- Classical (explicit) representation: $\varphi(x) \rightarrow \mathcal{F}$
- Kernel (implicit) representation:
  - $K: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ (paired comparisons, symmetric function)
  - $\mathcal{S}$ represented by a symmetric matrix $\mathbf{K} = [K(x_i, x_j)]_{i,j} \in \mathbb{R}^{n \times n}$
Definition
A kernel function is a function $K(\cdot, \cdot)$ such that for all $x, z \in \mathcal{X}$, it satisfies $K(x, z) = \varphi(x) \cdot \varphi(z)$ where $\varphi(x)$ is a mapping from $\mathcal{X}$ to an (inner product or Hilbert) space $\mathcal{H}$.

Definition
The Gram (or kernel) matrix associated with the kernel function $K(\cdot, \cdot)$, evaluated on a finite subset of examples $X = \{x_1, \ldots, x_n\}, x_i \in \mathcal{X}$, is the matrix $K \in \mathbb{R}^{n \times n}$ such that

$$K_{i,j} = K(x_i, x_j).$$

The matrix $K$ is symmetric and positive definite by definition.
Advantages of using kernels

- Representation with kernel matrices has some advantages:
  - same algorithm for different typologies of data
  - modularity of the design of kernel and algorithms
  - the integration of different views is simpler
- The dimensionality of data depends on the number of objects and not from their vector dimensionality
- Comparison between objects can result computationally simpler than using the explicit object representation: kernel computation vs. dot product
Kernel Modularity

Modularity in the design/definition of the kernel (representation) and the learning algorithm used for model computation (classification, regression, ranking, etc.)

- n objects $(x_1, y_1), \ldots, (x_n, y_n)$
- Kernel $K(x_i, x_j)$
- Matrix $K$ (nxn)
- Algorithm $w = A(K,Y)$
- Output $f_w(x)$
The Kernel Trick

Any algorithm for vectorial data which can be expressed in terms of dot-products can be implicitly executed in the feature space associated to a kernel, by simply replacing dot-products with kernel evaluations.

- **Kernelization** of popular linear or distance-based methods (e.g. Perceptron and kNN)

- Application of algorithms for vectorial data (SVM, Perceptron, etc.) to **non-vectorial data** using ad-hoc kernels (e.g. kernel for structures)
The Kernel Trick - Distances in feature space

Given two objects, \( x, z \in \mathcal{X} \), the distance between the two objects in feature space is computed by:

\[
d(x, z) = ||\varphi(x) - \varphi(z)||
\]

\[
d^2(x, z) = ||\varphi(x) - \varphi(z)||^2
\]

\[
= \varphi(x) \cdot \varphi(x) + \varphi(z) \cdot \varphi(z) - 2\varphi(x) \cdot \varphi(z)
\]

\[
= K(x, x) + K(z, z) - 2K(x, z)
\]

That is, \( d(x, z) = \sqrt{K(x, x) + K(z, z) - 2K(x, z)} \).

Note that the values \( \varphi(x), \varphi(z) \) are not explicitly used!
Linear vs. Poly kernel

Linear SVM

Poly SVM deg2
RBF kernels

\[ K(x, z) = \exp(-\gamma \|x - z\|^2) \]
Closure Properties

Let $K_1, K_2$ be kernels defined on $\mathcal{X} \times \mathcal{X}$. $a \in \mathbb{R}^+$, $\phi : \mathcal{X} \to \mathbb{R}^N$ with $K_3$ a kernel over $\mathbb{R}^N \times \mathbb{R}^N$. Then,

- $K(x, z) = K_1(x, z) + K_2(x, z)$ is a kernel
- $K(x, z) = aK_1(x, z)$ is a kernel
- $K(x, z) = K_1(x, z) \cdot K_2(x, z)$ is a kernel
- $K(x, z) = K_3(\phi(x), \phi(z))$ is a kernel

A kernel can be easily normalized (such to have normalized data in feature space $\|\phi(x)\| = 1$):

$$\tilde{K}(x, z) = \frac{K(x, z)}{\sqrt{K(x, x)K(z, z)}}$$
Kernel extensions to other types of inputs

- **Kernel for strings**
  Idea: given two strings, compute the number of shared sub-strings (dynamic programming algorithms exist to make efficient the computation of these kernels)

- **Kernel for trees**
  Idea: given two trees, compute the number of shared sub-trees (also here dynamic programming algorithms exist to make efficient the computation of these kernels)

- **Kernel for graphs**
  Idea: similar to the ones above, e.g. counting common walks.
Regression with SVM: Basic idea

When considering a regression problem, the idea is to define an $\epsilon$-tube: predictions which differ from the desired value for more than $\epsilon$ in absolute error are linearly penalized, otherwise they are not considered errors.
Regression with SVM

The $\epsilon$-tube idea leads to the following formulation

$$\min_{w, b, \xi, \xi^*} \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)$$
subject to:
$$\forall i \in \{1, \ldots, n\}$$
$$y_i - w \cdot x_i - b \leq \epsilon + \xi_i$$
$$w \cdot x_i + b - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

which has the following dual formulation...

$$\max_{\alpha, \alpha^*} -\epsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*) +$$
$$-\frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i, x_j)$$
subject to:
$$\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0$$
$$\alpha_i, \alpha_i^* \in [0, C]$$
Several kernel methods, including SVM, can be interpreted as solving the following problem:

$$\arg\min_{f \in \mathcal{H}} L(f(x_1), \ldots, f(x_n)) + \Lambda \|f\|_{\mathcal{H}}$$

- $L$ is a loss (or cost) function associated to the empirical risk.
- The norm is the “smoothness” of the function. In fact, the meaning of “smoothness” depends on the considered kernel and feature space.
- $\Lambda$ is a trade-off regularization coefficient.

The problem above can be shown to always have a solution of type:

$$f(x) = \mathbf{w} \cdot \varphi(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x)$$

That is the optimization problem can be formulated with $n$ variables. If $n \ll d$ then we get a computational advantage.
Disadvantages of kernels are:

- Memory space required to store the kernel matrix
- Time required to evaluate the scoring function (real-time applications)

Recently, there have been effort to ”linearize” the kernel. This can be done by sampling a finite set of non-linear features $\bar{\varphi}(x)$, such that

$$\langle \bar{\varphi}(x), \bar{\varphi}(z) \rangle \approx \langle \varphi(x), \varphi(z) \rangle = K(x, z)$$
Recap

- Motivate the Support Vector Machine (SVM) method from a theoretical point of view
- Discuss methods for handling non-linearly separable data with SVMs
- In the context of kernel methods (e.g. SVM), describe the concept of kernel and its close link with data representation
- Relationship between Perceptron and SVM: is it possible to see Perceptron as a kernel method?
- Relations between Neural Networks and SVM: where is the ”kernel” in Neural Networks? What are the differences in terms of objective function between the two methods?