Foundation of Intelligent Systems, Part I

SVM’s & Kernel Methods

mcuturi@i.kyoto-u.ac.jp
Support Vector Machines
The linearly-separable case
A criterion to select a linear classifier: the margin?
A criterion to select a linear classifier: the margin?
A criterion to select a linear classifier: the margin?
A criterion to select a linear classifier: the margin?
A criterion to select a linear classifier: the margin?
Largest Margin Linear Classifier?
Support Vectors with Large Margin
In equations

• The **training set** is a finite set of \( n \) data/class pairs:

\[
\mathcal{T} = \{(x_1, y_1), \ldots, (x_N, y_N)\},
\]

where \( x_i \in \mathbb{R}^d \) and \( y_i \in \{-1, 1\} \).

• We assume (for the moment) that the data are **linearly separable**, i.e., that there exists \((w, b) \in \mathbb{R}^d \times \mathbb{R}\) such that:

\[
\begin{align*}
\text{if } y_i = 1, \quad &w^T x_i + b > 0, \\
\text{if } y_i = -1, \quad &w^T x_i + b < 0.
\end{align*}
\]
How to find the largest separating hyperplane?

For the linear classifier \( f(x) = w^T x + b \) consider the *interstice* defined by the hyperplanes

- \( f(x) = w^T x + b = +1 \)
- \( f(x) = w^T x + b = -1 \)
The margin is $2/\|w\|$ 

- Indeed, the points $x_1$ and $x_2$ satisfy:

$$\begin{cases} w^T x_1 + b = 0, \\ w^T x_2 + b = 1. \end{cases}$$

- By subtracting we get $w^T (x_2 - x_1) = 1$, and therefore:

$$\gamma = 2\|x_2 - x_1\| = \frac{2}{\|w\|}.$$ 

where $\gamma$ is the margin.
All training points should be on the appropriate side

• For positive examples \((y_i = 1)\) this means:

\[
\mathbf{w}^T \mathbf{x}_i + b \geq 1
\]

• For negative examples \((y_i = -1)\) this means:

\[
\mathbf{w}^T \mathbf{x}_i + b \leq -1
\]

• in both cases:

\[
\forall i = 1, \ldots, n, \quad y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1
\]
Finding the optimal hyperplane

Finding the optimal hyperplane is equivalent to finding \((w, b)\) which minimize:

\[ \|w\|^2 \]

under the constraints:

\[ \forall i = 1, \ldots, n, \quad y_i (w^T x_i + b) - 1 \geq 0. \]

This is a classical quadratic program on \(\mathbb{R}^{d+1}\)

**linear constraints - quadratic objective**
Lagrangian

- In order to minimize:
  \[ \frac{1}{2} \| w \|^2 \]

  under the constraints:

  \[ \forall i = 1, \ldots, n, \quad y_i (w^T x_i + b) - 1 \geq 0. \]

- introduce one dual variable \( \alpha_i \) for each constraint,
- one constraint for each training point.
- the Lagrangian is, for \( \alpha \geq 0 \) (that is for each \( \alpha_i \geq 0 \))

  \[ L(w, b, \alpha) = \frac{1}{2} \| w \|^2 - \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1). \]
The Lagrange dual function

\[ g(\alpha) = \inf_{w \in \mathbb{R}^d, b \in \mathbb{R}} \left\{ \frac{1}{2} \| w \|^2 - \sum_{i=1}^{n} \alpha_i \left( y_i (w^T x_i + b) - 1 \right) \right\} \]

the saddle point conditions give us that at the minimum in \( w \) and \( b \)

\[ w = \sum_{i=1}^{n} \alpha_i y_i x_i, \quad \text{(derivating w.r.t w)} \quad (*) \]

\[ 0 = \sum_{i=1}^{n} \alpha_i y_i, \quad \text{(derivating w.r.t b)} \quad (**) \]

substituting (*) in \( g \), and using (**) as a constraint, get the dual function \( g(\alpha) \).

- To solve the dual problem, maximize \( g \) w.r.t. \( \alpha \).
- **Strong duality holds** : primal and dual problems have the same optimum.
- KKT gives us \( \alpha_i (y_i (w^T x_i + b) - 1) = 0 \),
  ...hence, either \( \alpha_i = 0 \) or \( y_i (w^T x_i + b) = 1 \).
- \( \alpha_i \neq 0 \) only for points on the support hyperplanes \( \{(x, y)| y_i (w^T x_i + b) = 1\} \).
The dual problem is thus

\[
\text{maximize } g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{such that } \alpha \succeq 0, \sum_{i=1}^{n} \alpha_i y_i = 0.
\]

This is a \textbf{quadratic program} in $\mathbb{R}^n$, with \textit{box constraints}. $\alpha^*$ can be computed using optimization software (e.g. built-in \texttt{matlab} function)
Recovering the optimal hyperplane

- With $\alpha^*$, we recover $(w^T, b^*)$ corresponding to the **optimal hyperplane**.
- $w^T$ is given by $w^T = \sum_{i=1}^{n} y_i \alpha_i x_i^T$,
- $b^*$ is given by the conditions on the support vectors $\alpha_i > 0$, $y_i (w^T x_i + b) = 1$,

$$b^* = -\frac{1}{2} \left( \min_{y_i=1, \alpha_i > 0} (w^T x_i) + \max_{y_i=-1, \alpha_i > 0} (w^T x_i) \right)$$

- the **decision function** is therefore:

$$f^*(x) = w^T x + b^*$$

$$= \left( \sum_{i=1}^{n} y_i \alpha_i x_i^T \right) x + b^*.$$

- Here the **dual** solution gives us directly the **primal** solution.
Interpretation: support vectors

\[ \alpha = 0 \]

\[ \alpha > 0 \]
Another interpretation: Convex Hulls

go back to 2 sets of points that are linearly separable
Another interpretation: Convex Hulls

Linearly separable = convex hulls do not intersect
Another interpretation: Convex Hulls

Find two closest points, one in each convex hull
Another interpretation: Convex Hulls

The SVM = bisection of that segment
Another interpretation: Convex Hulls

support vectors = extreme points of the faces on which the two points lie
The non-linearly separable case

(when convex hulls intersect)
What happens when the data is not linearly separable?
What happens when the data is not linearly separable?
What happens when the data is not linearly separable?
What happens when the data is not linearly separable?
Soft-margin SVM?

- Find a trade-off between **large margin** and **few errors**.

- Mathematically:

\[
\min_f \left\{ \frac{1}{\text{margin}(f)} + C \times \text{errors}(f) \right\}
\]

- \(C\) is a parameter
Soft-margin SVM formulation?

- The **margin** of a labeled point \((x, y)\) is
  \[
  \text{margin}(x, y) = y \left( w^T x + b \right)
  \]

- The **error** is
  - 0 if \(\text{margin}(x, y) > 1\),
  - \(1 - \text{margin}(x, y)\) otherwise.

- The soft margin SVM solves:
  \[
  \min_{w, b} \left\{ \|w\|^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i \left( w^T x_i + b \right) \} \right\}
  \]

- \(c(u, y) = \max\{0, 1 - yu\}\) is known as the **hinge loss**.

- \(c(w^T x_i + b, y_i)\) associates a mistake cost to the decision \(w, b\) for example \(x_i\).
Dual formulation of soft-margin SVM

- The soft margin SVM program

\[
\min_{w,b}\{\|w\|^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i (w^T x_i + b)\}\}
\]

can be rewritten as

\[
\begin{align*}
\text{minimize} & \quad \|w\|^2 + C \sum_{i=1}^{n} \xi_i \\
\text{such that} & \quad y_i (w^T x_i + b) \geq 1 - \xi_i
\end{align*}
\]

- In that case the dual function

\[
g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j,
\]

which is finite under the constraints:

\[
\begin{cases}
0 \leq \alpha_i \leq C, & \text{for } i = 1, \ldots, n \\
\sum_{i=1}^{n} \alpha_i y_i = 0.
\end{cases}
\]
Interpretation: bounded and unbounded support vectors

\[ \alpha = 0 \]

\[ \alpha = C \]

\[ 0 < \alpha < C \]
What about the convex hull analogy?

• Remember the separable case

• Here we consider the case where the two sets are not linearly separable, i.e. their convex hulls intersect.
What about the convex hull analogy?

**Definition 1.** Given a set of $n$ points $\mathcal{A}$, and $0 \leq C \leq 1$, the set of finite combinations

$$\sum_{i=1}^{n} \lambda_i x_i, 1 \leq \lambda_i \leq C, \sum_{i=1}^{n} \lambda_i = 1,$$

is the $(C)$ reduced convex hull of $\mathcal{A}$

- Using $C = 1/2$, the reduced convex hulls of $\mathcal{A}$ and $\mathcal{B}$,

- Soft-SVM with $C =$ closest two points of $C$-reduced convex hulls.
Kernels
Kernel trick for SVM’s

- use a mapping $\phi$ from $X$ to a feature space,
- which corresponds to the kernel $k$:

$$\forall x, x' \in X, \quad k(x, x') = \langle \phi(x), \phi(x') \rangle$$

- Example: if $\phi(x) = \phi \left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix}$, then

$$k(x, x') = \langle \phi(x), \phi(x') \rangle = (x_1)^2(x_1')^2 + (x_2)^2(x_2')^2.$$
Training a SVM in the feature space

Replace each $x^T x'$ in the SVM algorithm by $\langle \phi(x), \phi(x') \rangle = k(x, x')$

- **Reminder**: the dual problem is to maximize

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

under the constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0. \end{cases}$$

- The **decision function** becomes:

$$f(x) = \langle w, \phi(x) \rangle + b^*$$

$$= \sum_{i=1}^{n} y_i \alpha_i k(x_i, x) + b^*. \quad (1)$$
The Kernel Trick?

The explicit computation of $\phi(x)$ is not necessary. The kernel $k(x, x')$ is enough.

- the SVM optimization for $\alpha$ works **implicitly** in the feature space.
- the SVM is a kernel algorithm: only need to input $K$ and $y$:

\[
\text{maximize } g(\alpha) = \alpha^T 1 - \frac{1}{2} \alpha^T (K \odot yy^T) \alpha \\
\text{such that } 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, n \\
\sum_{i=1}^n \alpha_i y_i = 0.
\]

- $K$'s **positive definite** $\iff$ problem has an unique optimum
- the decision function is $f(\cdot) = \sum_{i=1}^n \alpha_i k(x_i, \cdot) + b$. 

FIS - 2012
Kernel example: polynomial kernel

- For $\mathbf{x} = (x_1, x_2)^\top \in \mathbb{R}^2$, let $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$:

$$K(\mathbf{x}, \mathbf{x}') = x_1^2x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2$$

$$= \{x_1x_1' + x_2x_2'\}^2$$

$$= \{\mathbf{x}^T\mathbf{x}'\}^2.$$
Kernels are Trojan Horses onto Linear Models

- With kernels, complex structures can enter the realm of linear models
What is a kernel

A kernel \( k \) is a function

\[
k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \quad (x, y) \rightarrow k(x, y)
\]

which compares two objects of a space \( \mathcal{X} \), e.g.:

- strings, texts and sequences,
- images, audio and video feeds,
- graphs, interaction networks and 3D structures

whatever actually... time-series of graphs of images? graphs of texts?...
Fundamental properties of a kernel

**symmetric**

\[ k(x, y) = k(y, x). \]

**positive-(semi)definite**

for any *finite* family of points \( x_1, \ldots, x_n \) of \( \mathcal{X} \), the matrix

\[
K = \begin{bmatrix}
  k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_i) & \cdots & k(x_1, x_n) \\
  k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_i) & \cdots & k(x_2, x_n) \\
  \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\
  k(x_i, x_1) & k(x_i, x_2) & \cdots & k(x_i, x_i) & \cdots & k(x_i, x_n) \\
  \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\
  k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_i) & \cdots & k(x_n, x_n)
\end{bmatrix}
\]

is positive semidefinite (has a nonnegative spectrum).

\( K \) is often called the **Gram matrix** of \( \{x_1, \ldots, x_n\} \) using \( k \).
What can we do with a kernel?
The setting

• Pretty simple setting: a set of objects \( x_1, \ldots, x_n \) of \( \mathcal{X} \)

• Sometimes additional information on these objects
  
  ○ labels \( y_i \in \{-1, 1\} \) or \( \{1, \ldots, \#(\text{classes})\} \),
  ○ scalar values \( y_i \in \mathbb{R} \),
  ○ associated object \( y_i \in \mathcal{Y} \)

• A kernel \( k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R} \).
Important concepts and perspectives

- The functional perspective: represent **points as functions**.
- **Nonlinearity**: linear combination of kernel evaluations.
- Summary of a sample through its **kernel matrix**.
Represent any point in $\mathcal{X}$ as a function

For every $x$, the map

$$\begin{align*}
  x &\mapsto k(x, \cdot)
\end{align*}$$

associates to $x$ a function $k(x, \cdot)$ from $\mathcal{X}$ to $\mathbb{R}$.

- Suppose we have a kernel $k$ on bird images

- Suppose for instance

$$k\left(\begin{array}{c}
  \text{bird 1} \\
  \text{bird 2}
\end{array}\right) = .32$$
Represent any point in $\mathcal{X}$ as a function

- We examine one image in particular:
- With kernels, we get a representation of that bird as a real-valued function, defined on the space of birds, represented here as $\mathbb{R}^2$ for simplicity.

schematic plot of $k\left(\vec{\text{bird}}, \cdot \right)$. 
Represent any point in $\mathcal{X}$ as a function

- If the bird example was confusing...

- $k\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \end{bmatrix}\right) = \left(\begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + .3\right)^2$

- From a point in $\mathbb{R}^2$ to a function defined over $\mathbb{R}^2$.

- We assume implicitly that the **functional representation** will be more useful than the **original representation**.
Decision functions as linear combination of kernel evaluations

- Linear decisions functions are a major tool in statistics, that is functions

\[ f(x) = \beta^T x + \beta_0. \]

- Implicitly, a point \( x \) is processed depending on its characteristics \( x_i \),

\[ f(x) = \sum_{i=1}^{d} \beta_i x_i + \beta_0. \]

the free parameters are scalars \( \beta_0, \beta_1, \ldots, \beta_d \).

- Kernel methods yield candidate decision functions

\[ f(x) = \sum_{j=1}^{n} \alpha_j k(x_j, x) + \alpha_0. \]

the free parameters are scalars \( \alpha_0, \alpha_1, \ldots, \alpha_n \).
Decision functions as linear combination of kernel evaluations

- linear decision surface / linear expansion of kernel surfaces (here $k_G(x_i, \cdot)$)

- Kernel methods are considered non-linear tools.
- Yet not completely “nonlinear” → only one-layer of nonlinearity.

Kernel methods use the data as a functional base to define decision functions.
Decision functions as linear combination of kernel evaluations

\[ f(x) = \sum_{i=1}^{N} \alpha_i \ k(x_i, x) \]

database \( \{x_i, i = 1, \ldots, N\} \)

- \( f \) is any predictive function of interest of a new point \( x \).
- Weights \( \alpha \) are optimized with a kernel machine (e.g. support vector machine)

intuitively, kernel methods provide decisions based on how similar a point \( x \) is to each instance of the training set
The Gram matrix perspective

- Imagine a little task: you have read 100 novels so far.

- You would like to know whether you will enjoy reading a new novel.

- A few options:
  - read the book...
  - have friends read it for you, read reviews.
  - try to guess, based on the novels you read, if you will like it
The Gram matrix perspective

Two distinct approaches

• Define what **features** can characterize a book.
  
  ○ Map each book in the library onto vectors

  \[ \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \]

  typically the \( x_i \)'s can describe...

  ▶ # pages, language, year 1st published, country,
  ▶ coordinates of the main action, keyword counts,
  ▶ author’s prizes, popularity, booksellers ranking

• Challenge: find a decision function using 100 ratings and features.
The Gram matrix perspective

- Define what makes two novels similar,
  - Define a kernel $k$ which quantifies novel similarities.
  - Map the library onto a Gram matrix

$$\rightarrow K = \begin{bmatrix}
    k(b_1, b_1) & k(b_1, b_2) & \cdots & k(b_1, b_{100}) \\
    k(b_2, b_1) & k(b_2, b_2) & \cdots & k(b_2, b_{100}) \\
    \vdots & \vdots & \ddots & \vdots \\
    k(b_n, b_1) & k(b_n, b_2) & \cdots & k(b_{100}, b_{100}) 
\end{bmatrix}$$

- Challenge: find a decision function that takes this $100 \times 100$ matrix as an input.
The Gram matrix perspective

Given a new novel,

• with the **features approach**, the prediction can be rephrased as **what are the features of this new book?** what **features** have I found in the past that were good indicators of my taste?

• with the **kernel approach**, the prediction is rephrased as **which novels this book is similar or dissimilar to?** what **pool of books** did I find the most influentials to define my tastes accurately?

**kernel methods only use kernel similarities**, do not consider features.

Features can help define similarities, but **never considered elsewhere**.
The Gram matrix perspective

in kernel methods, clear separation between the kernel...

and Convex optimization (thanks to psdness of $K$, more later) to output the $\alpha$'s.