Lecture 13: Kernel Methods

Introduction to Machine Learning [25737]

Sajjad Amini

Sharif University of Technology

Contents

References

Except explicitly cited, the reference for the material in slides is:

- Murphy, K. P. (2022). *Probabilistic machine learning: an introduction.* MIT press.

Section 1

Approach Definition

Approach Definition

Previous Methods

Till now, we use the following procedure to evaluate the output y given input vector $\boldsymbol{x}$:

- Considering a parameterized model with parameter vector $\boldsymbol{\theta}$
- Estimating parameters using dataset $\{(\boldsymbol{x}_n, y_n)\}$ $(\widehat{\boldsymbol{\theta}})$
  - Plug-in approximation
  - Posterior distribution calculation
- Evaluating y using assumed model
  - Evaluating using plug-in approximation (1 model)
  - Evaluating using posterior predictive distribution (All possible models)

Approach Definition

Kernel Method

Assume we have dataset $\{(\boldsymbol{x}_n, y_n)\}$ and unknown function f that $y = f(\boldsymbol{x})$, then in kernel method, the procedure is:

- Evaluate a similarity between query vector $\boldsymbol{x}_q$ and all of input training vectors $\{\boldsymbol{x}_i\}_{i=1}^{N}$
- Use the measures similarity as weights to generate $f(\boldsymbol{x}_q)$ based on $\{f(\boldsymbol{x}_i)\}_{i=1}^{N}$

Note that *Kernels* are special functions that determine the similarity used for $f(\boldsymbol{x}_q)$ estimation.

Kernel Methods

- Kernel methods are nonparametric
- In model based methods, we compress the dataset information into a fixed length vector $\widehat{\boldsymbol{\theta}}$, while in kernel method we need dataset to estimate $f(\boldsymbol{x}_q)$

Section 2

Mercer Kernel

Mercer Kernel [1]

Kernel

Assume an abstract space $\mathcal{X}$. Then function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel function. Kernel function usually (not necessarily) have the following properties:

$$\begin{aligned} &\text{Symmetry:} \quad \forall \boldsymbol{x}, \boldsymbol{x}' : \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x}', \boldsymbol{x}) \\ &\text{Positivity:} \quad \forall \boldsymbol{x}, \boldsymbol{x}' : \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \geq 0, (\boldsymbol{x} \neq \boldsymbol{x}') \end{aligned}$$

Mercer (positive definite) Kernel

A symmetric kernel $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is Mercer (PSD) kernel if:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) c_i c_j \geq 0$$

for any finite set of N distinct samples from $\mathcal{X} \supseteq X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and any choice of numbers $c_i \in \mathbb{R}$.

Mercer Kernel

Gram matrix for N Datapoints

Given N datapoints $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ and symmetric kernel $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the Gram matrix is:

$$\boldsymbol{K} = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots \\ \mathcal{K}(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & \mathcal{K}(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

Mercer Kernel and Gram Matrix

Symmetric positive kernel $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is Mercer iff the Gram matrix is positive definite for any finite set of N distinct samples from $\mathcal{X} \supseteq X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$.

Hilbert Space

Inner Product

Let $\mathcal{H}$ be a vector space over $\mathbb{R}$. A function $\langle \cdot, \cdot \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is said to be an inner product on $\mathcal{H}$ if:

1. $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle_{\mathcal{H}} = \alpha_1 \langle f_1, g \rangle_{\mathcal{H}} + \alpha_2 \langle f_2, g \rangle_{\mathcal{H}}$
2. $\langle f, g \rangle_{\mathcal{H}} = \langle g, f \rangle_{\mathcal{H}}$
3. $\langle f, f \rangle_{\mathcal{H}} \geq 0$ and $\langle f, f \rangle_{\mathcal{H}} = 0$ if and only if $f = 0$

Norm

Using inner product, we can define norm as: $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$

Hilbert Space

A Hilbert space is a vector space on which an inner product is defines (along with other technical conditions)

Checking Positive Definiteness of Kernel

Mercer (positive definite) Kernel

Let $\mathcal{H}$ be any Hilbert space, $\mathcal{X}$ a non-empty set and $\phi : \mathcal{X} \to \mathcal{H}$. Then kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$ is positive definite.
Proof:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) c_i c_j = \sum_{i=1}^{N}\sum_{j=1}^{N} \langle c_i \phi(\boldsymbol{x}_i), c_j \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}} = \left\| \sum_{i=1}^{N} c_i \phi(\boldsymbol{x}_i) \right\|_{\mathcal{H}}^2 \geq 0$$

Simple Mercer Kernel

Show that kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}', \boldsymbol{x} \in \mathbb{R}^n$ is Mercer.
Solution: We can introduce Hilbert space $\mathcal{H} = \mathbb{R}^n$ with the definition $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle = \boldsymbol{x}^T \boldsymbol{x}'$ for inner product and mapping $\phi(\boldsymbol{x}) = \boldsymbol{x}$. Thus $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$ is a Mercer kernel.

Mercer's Theorem

Mercer's Theorem

Assume Gram matrix $\boldsymbol{K}$ to be positive definite. Then from eigen decomposition we have $\boldsymbol{K} = \boldsymbol{U}^T \boldsymbol{\Lambda} \boldsymbol{U}$ where:

$$\begin{aligned} \boldsymbol{\Lambda} &= \operatorname{diag}(\lambda_1, \dots, \lambda_N), \lambda_i > 0 \text{ for } i = 1, \dots, N \\ \boldsymbol{U} &= [\boldsymbol{u}_1, \dots, \boldsymbol{u}_N] \end{aligned}$$

We can rewrite $\boldsymbol{K} = (\boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{U})^T (\boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{U}) = \widehat{\boldsymbol{U}}^T \widehat{\boldsymbol{U}}$ where $\widehat{\boldsymbol{U}} = [\widehat{\boldsymbol{u}}_1, \dots, \widehat{\boldsymbol{u}}_N]$. Thus:

$$\boldsymbol{K} = \begin{bmatrix} - & \widehat{\boldsymbol{u}}_1^T & - \\ & \vdots & \\ - & \widehat{\boldsymbol{u}}_N^T & - \end{bmatrix} \begin{bmatrix} | & & | \\ \widehat{\boldsymbol{u}}_1 & \cdots & \widehat{\boldsymbol{u}}_N \\ | & & | \end{bmatrix} \Rightarrow k_{ij} = \widehat{\boldsymbol{u}}_i^T \widehat{\boldsymbol{u}}_j = \langle \widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j \rangle$$

So $\boldsymbol{\phi}(\boldsymbol{x}_i) = \widehat{\boldsymbol{u}}_i$ and we can write the entries in form of inner product.

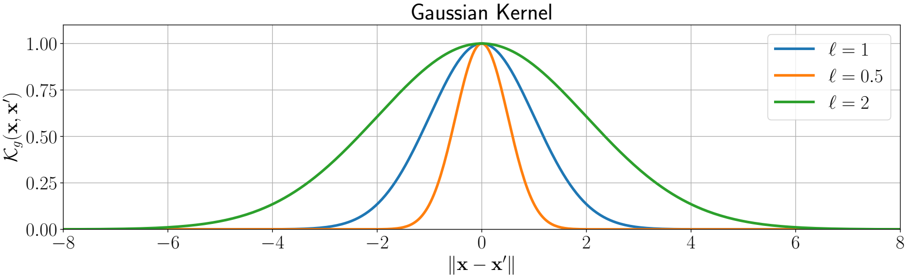
Popular Mercer Kernel [2]

Stationary Kernels

Kernels of the form $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\|\boldsymbol{x} - \boldsymbol{x}'\|)$ are called stationary kernels.

Gaussian (exponentiated quadratic) kernel

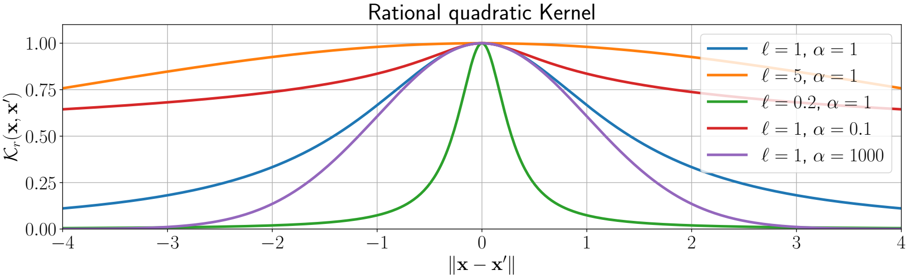
$$\mathcal{K}_g(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2l^2}\right), \; l : \text{bandwidth}$$



Popular Mercer Kernel [2]

Rational quadratic kernel

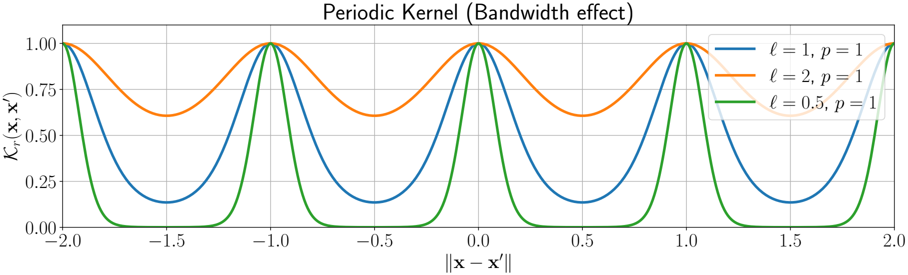
$$\mathcal{K}_r(\boldsymbol{x}, \boldsymbol{x}') = \left(1 + \frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\alpha l^2}\right)^{-\alpha}, \quad \begin{cases} l : \text{bandwidth} \\ \alpha : \text{scale} \end{cases}$$



Popular Mercer Kernel [2]

Periodic kernel

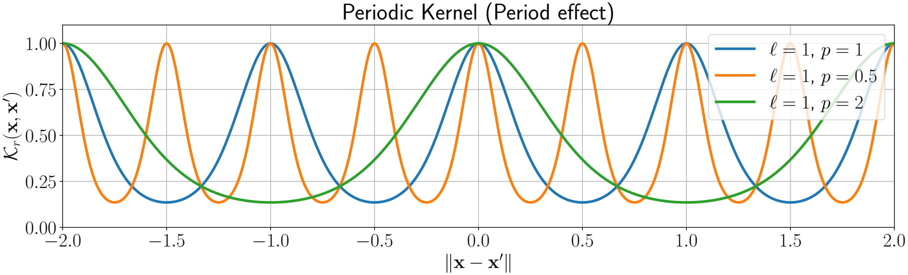
$$\mathcal{K}_p(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{2}{l^2}\sin^2\left(\pi\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|}{p}\right)\right), \quad \begin{cases} l : \text{bandwidth} \\ \alpha : \text{period} \end{cases}$$



Popular Mercer Kernel [2]

Periodic kernel

$$\mathcal{K}_p(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{2}{l^2}\sin^2\left(\pi\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|}{p}\right)\right), \quad \begin{cases} l : \text{bandwidth} \\ \alpha : \text{period} \end{cases}$$



Section 3

Gaussian Processes

Gaussian Processes

Gaussian Processes

Gaussian processes (GP) are an approach to defining distribution over functions of the form $f : \mathcal{X} \to \mathbb{R}$. To this end, we assume $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_M)]^T$ to be jointly Gaussian for any $M > 0$ and with:

- $\boldsymbol{\mu} = [m(\boldsymbol{x}_1), \ldots, m(\boldsymbol{x}_M)]^T$
- $\Sigma_{ij} = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ or equivalently $\boldsymbol{\Sigma} = \boldsymbol{K}$

To use the above distrinution, we can consider the special case $M = N + 1$ and joint distribution $p([f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N), f(\boldsymbol{x}_q)]^T)$ and infer $f(\boldsymbol{x}_q)$.

Covariance matrix

A valid covariance matrix must be symmtric and positive semi-definite. These conditions are already met in $\boldsymbol{K}$ for a Mercer kernel.

Prior Information

We can assume the mean function $m(\cdot)$ and covariance generative kernel $\mathcal{K}(\cdot, \cdot)$ to be the prior information.

Sampling from Prior Information

Sampling from Prior Information

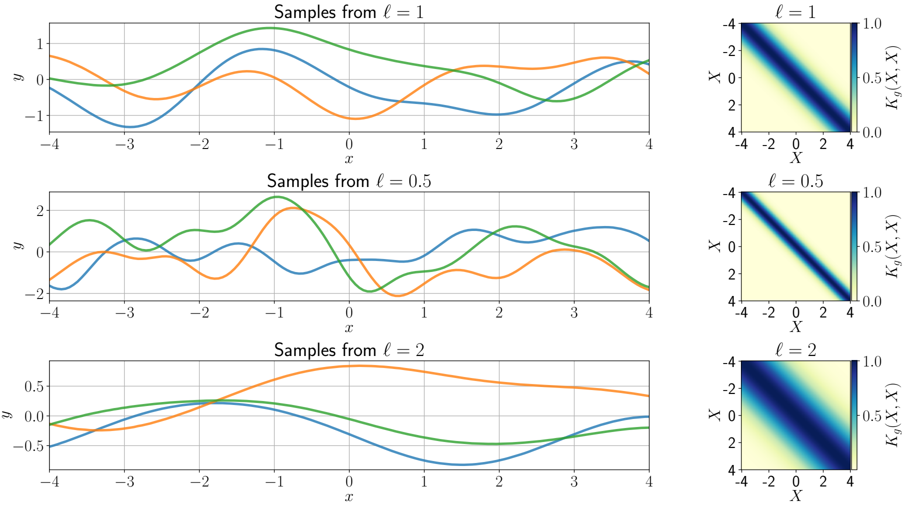
Consider $\boldsymbol{x} \in \mathbb{R}^D$, $m(\boldsymbol{x})$ and a Mercer kernel $\mathcal{K}(\cdot, \cdot)$. Then for sampling from the prior we have the following steps:

- Assume a set of $\boldsymbol{x}$ values where we want to evaluate the sample as $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S\}$.
- Generate the mean vector and covariance matrix as:

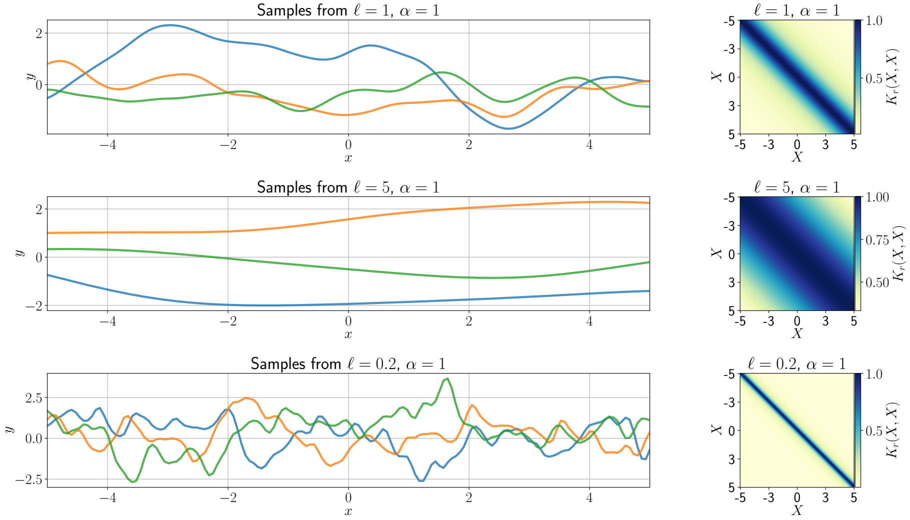
$$\boldsymbol{\mu} = \begin{bmatrix} m(\boldsymbol{x}_1) \\ m(\boldsymbol{x}_2) \\ \vdots \\ m(\boldsymbol{x}_S) \end{bmatrix}, \ \boldsymbol{\Sigma} = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_1) & \ldots & \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_S) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\boldsymbol{x}_S, \boldsymbol{x}_1) & \ldots & \mathcal{K}(\boldsymbol{x}_S, \boldsymbol{x}_S) \end{bmatrix}$$

- Sample vector $\boldsymbol{y} = [y_1, \ldots, y_S] \in \mathbb{R}^S$
- The realization of the function at evaluation points is $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_S, y_S)\}$

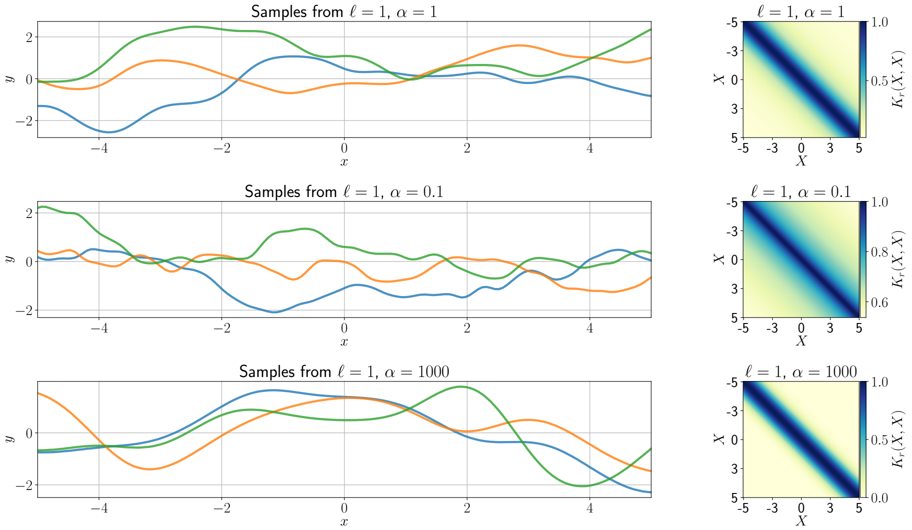
Sampling from Zero Mean and Gaussian Kernel [2]



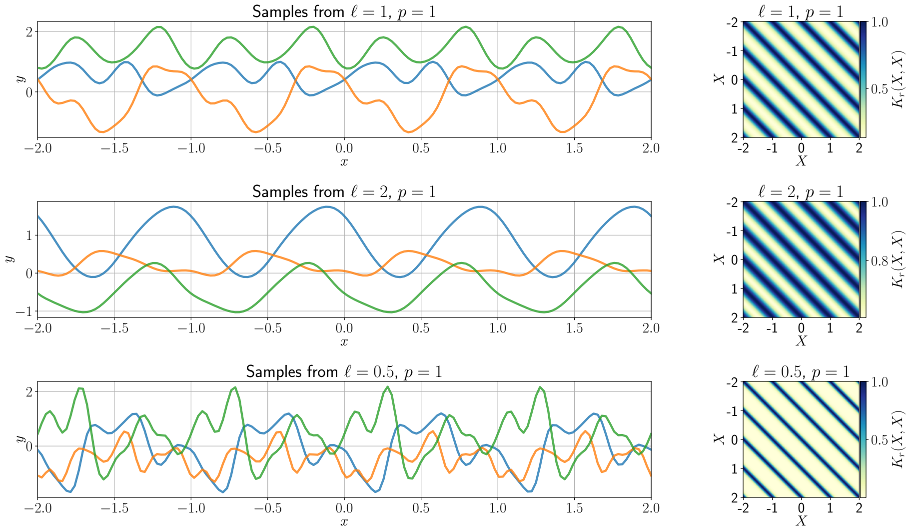
Sampling from Zero Mean and Rationale Kernel [2]



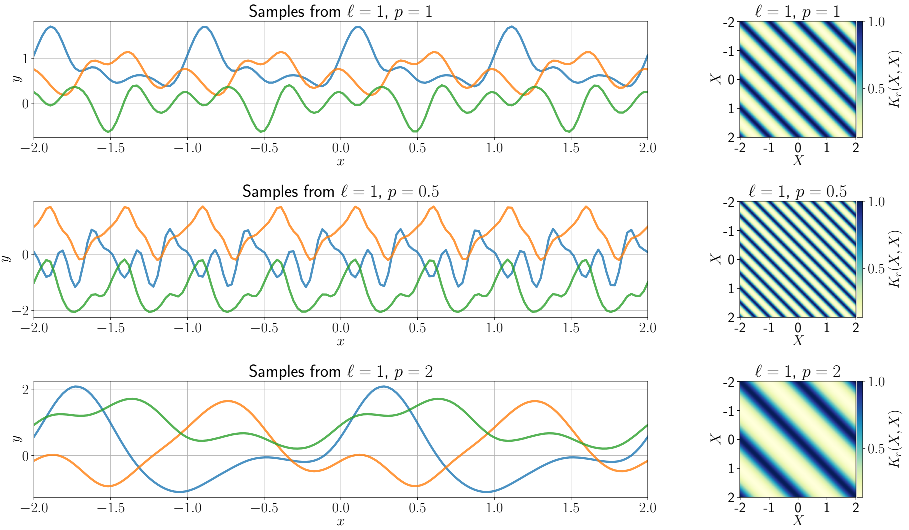
Sampling from Zero Mean and Rationale Kernel [2]



Sampling from Zero Mean and Periodic Kernel [2]



Sampling from Zero Mean and Periodic Kernel [2]



Posterior for Noise Free Observations

Noise Free Observations

Suppose we observe training dataset $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ where $y_n = f(\boldsymbol{x}_n)$ is noise-free observation and we have queries $\{\boldsymbol{x}_i^{(e)}\}_{i=1}^{N_\star}$. Assume:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}, \ \boldsymbol{X}_\star = \begin{bmatrix} \boldsymbol{x}_1^{(e)T} \\ \vdots \\ \boldsymbol{x}_{N_\star}^{(e)T} \end{bmatrix}, \ \boldsymbol{\mu}_{\boldsymbol{X}} = \begin{bmatrix} m(\boldsymbol{x}_1) \\ \vdots \\ m(\boldsymbol{x}_N) \end{bmatrix}, \ \boldsymbol{\mu}_\star = \begin{bmatrix} m(\boldsymbol{x}_1^{(e)}) \\ \vdots \\ m(\boldsymbol{x}_{N_\star}^{(e)}) \end{bmatrix}$$

$$\boldsymbol{f}_X = \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix}, \ \boldsymbol{f}_\star = \begin{bmatrix} f(\boldsymbol{x}_1^{(e)}) \\ \vdots \\ f(\boldsymbol{x}_{N_\star}^{(e)}) \end{bmatrix}, \ \begin{cases} \boldsymbol{K}_{X,X} = \mathcal{K}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{N \times N} \\ \boldsymbol{K}_{X,\star} = \mathcal{K}(\boldsymbol{X}, \boldsymbol{X}_\star) \in \mathbb{R}^{N \times N_\star} \\ \boldsymbol{K}_{\star,\star} = \mathcal{K}(\boldsymbol{X}_\star, \boldsymbol{X}_\star) \in \mathbb{R}^{N_\star \times N_\star} \end{cases}$$

Prior Information

Prior Based on GP

Based on GP, we have joint distribution $p(\boldsymbol{f}_X, \boldsymbol{f}_\star | \boldsymbol{X}, \boldsymbol{X}_\star)$ as:

$$\begin{bmatrix} \boldsymbol{f}_X \\ \boldsymbol{f}_\star \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{x}} \\ \boldsymbol{\mu}_\star \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{X,X} & \boldsymbol{K}_{X,\star} \\ \boldsymbol{K}_{X,\star}^T & \boldsymbol{K}_{\star,\star} \end{bmatrix} \right)$$

Posterior By MVN Conditionals

Assuming we have observe $\boldsymbol{f}_X$, we can compute the posterior over $\boldsymbol{f}_\star$ using MVN conditionals.

MVN Conditionals

Suppose $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix}$ is jointly Gaussian with $\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}$, $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$ Then the posterior conditional is given by $p(\boldsymbol{y}_1 | \boldsymbol{y}_2) = \mathcal{N}(\boldsymbol{y}_1 | \boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2})$ where:

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{y}_2 - \boldsymbol{\mu}_2),\ \boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}$$

Posterior for Noise Free Observations

Calculating Posterior

Using prior distribution and MVN conditionals, we conclue $p(\boldsymbol{f}_\star|\mathcal{D}, \boldsymbol{X}_\star) = \mathcal{N}(\boldsymbol{f}_\star|\boldsymbol{\mu}_\star^{post}, \boldsymbol{\Sigma}_\star^{post})$ where:

$$\boldsymbol{\mu}_\star^{post} = \boldsymbol{\mu}_\star + \boldsymbol{K}_{X,\star}^T \boldsymbol{K}_{X,X}^{-1}(\boldsymbol{f}_X - \boldsymbol{\mu}_X)$$
$$\boldsymbol{\Sigma}_\star^{post} = \boldsymbol{K}_{\star,\star} - \boldsymbol{K}_{X,\star}^T \boldsymbol{K}_{X,X}^{-1} \boldsymbol{K}_{X,\star}$$

Training Data Interpolator

Training Data Interpolator

Suppose we use zero mean prior ($m(\boldsymbol{x}) = 0$) and kernel $\mathcal{K}(\cdot, \cdot)$. Assume the following case:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}, \ \boldsymbol{X}_\star = \begin{bmatrix} \boldsymbol{x}_i^T \end{bmatrix}$$

Then we can show that:

$$\Rightarrow \boldsymbol{K}_{X,\star} = \boldsymbol{K}_{X,X}(:, i) \Rightarrow \boldsymbol{K}_{X,\star}^T \boldsymbol{K}_{X,X}^{-1} = \boldsymbol{K}_{X,X}(i, :) \boldsymbol{K}_{X,X}^{-1} = \boldsymbol{e}_i$$

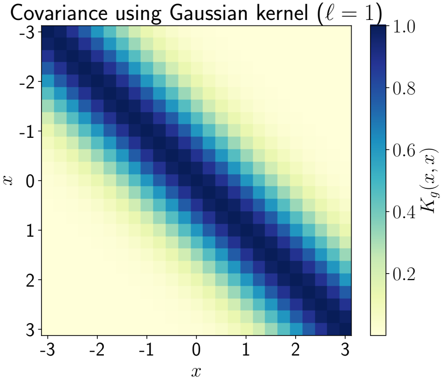
Using above equality, we have:

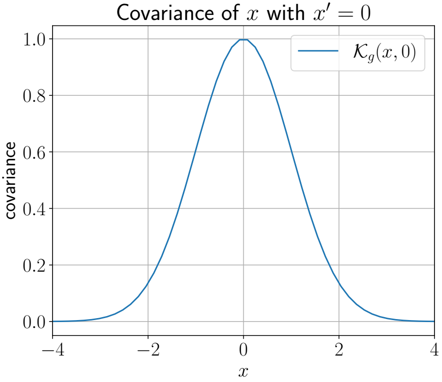
$$\begin{aligned} \boldsymbol{\mu}_\star^{post} =& \mu_\star + \boldsymbol{K}_{X,\star}^T \boldsymbol{K}_{X,X}^{-1} (\boldsymbol{f}_X - \boldsymbol{\mu}_X) = 0 + \boldsymbol{e}_i (\boldsymbol{f}_X - \boldsymbol{0}) = \boldsymbol{f}_X(i) = f(\boldsymbol{x}_i) \\ \boldsymbol{\Sigma}_\star^{post} =& \boldsymbol{K}_{\star,\star} - \boldsymbol{K}_{X,\star}^T \boldsymbol{K}_{X,X}^{-1} \boldsymbol{K}_{X,\star} = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_i) - \boldsymbol{e}_i \boldsymbol{K}_{X,\star} \\ =& \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_i) - \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_i) = 0 \end{aligned}$$

Example: Kernel Introduction [3]

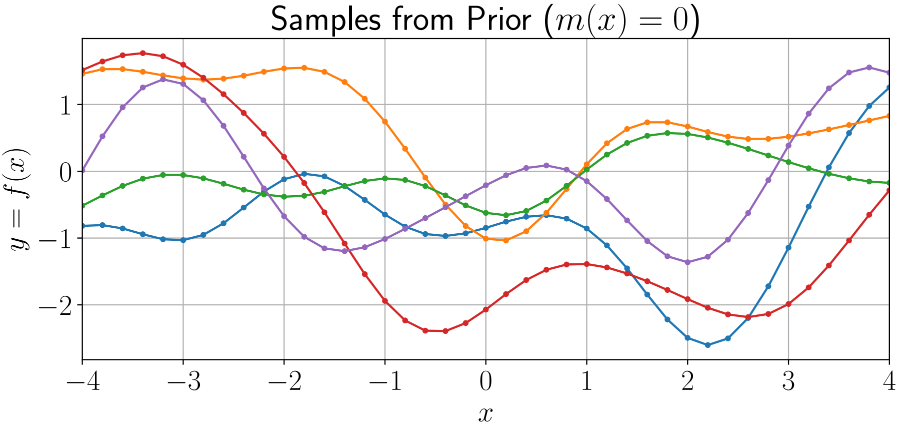
Gaussian Kernel

Assume Gussian kernel with unit bandwith as: $\mathcal{K}_g(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2}\right)$

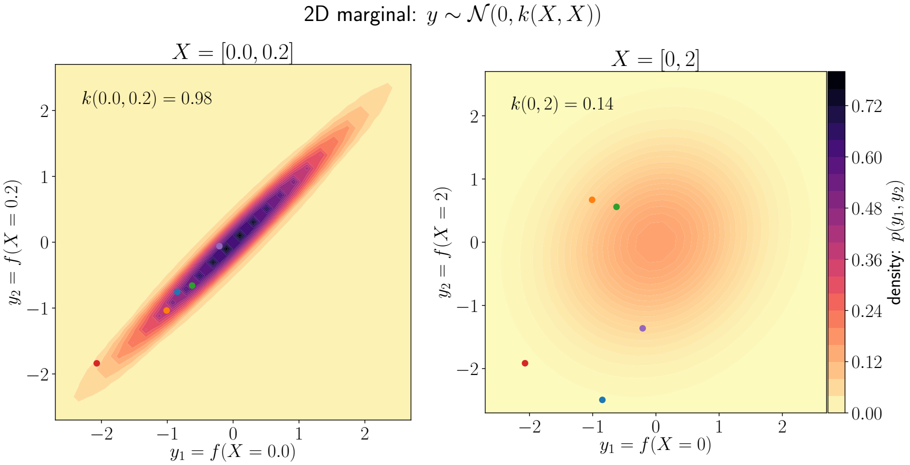




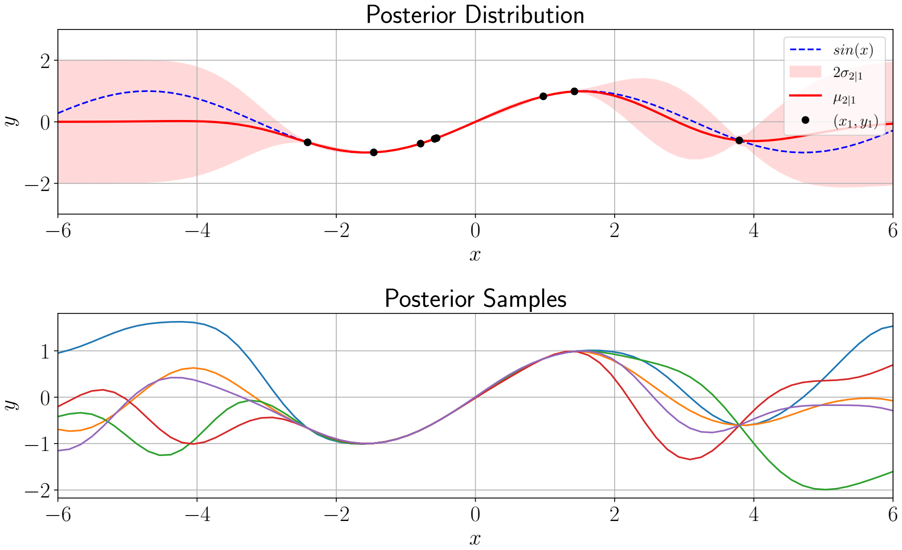
Example: Prior Samples [3]



Example: Prior Low Dimensional Samples [3]



Example: Posterior Distribution [3]



Noisy Observations

Suppose we observe training dataset $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ where $y_n = f(\boldsymbol{x}_n) + \epsilon_n$ ($\epsilon_n \sim \mathcal{N}(0, \sigma_y^2)$) is noisy observation and we have queries $\{\boldsymbol{x}_i^{(e)}\}_{i=1}^{N_\star}$. Assume:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}, \ \boldsymbol{X}_\star = \begin{bmatrix} \boldsymbol{x}_1^{(e)T} \\ \vdots \\ \boldsymbol{x}_{N_\star}^{(e)T} \end{bmatrix}, \ \boldsymbol{\mu}_{\boldsymbol{X}} = \begin{bmatrix} m(\boldsymbol{x}_1) \\ \vdots \\ m(\boldsymbol{x}_N) \end{bmatrix}, \ \boldsymbol{\mu}_\star = \begin{bmatrix} m(\boldsymbol{x}_1^{(e)}) \\ \vdots \\ m(\boldsymbol{x}_{N_\star}^{(e)}) \end{bmatrix}$$

$$\boldsymbol{y} = \begin{bmatrix} f(\boldsymbol{x}_1) + \epsilon_1 \\ \vdots \\ f(\boldsymbol{x}_N) + \epsilon_N \end{bmatrix}, \ \boldsymbol{f}_\star = \begin{bmatrix} f(\boldsymbol{x}_1^{(e)}) \\ \vdots \\ f(\boldsymbol{x}_{N_\star}^{(e)}) \end{bmatrix}, \ \begin{cases} \boldsymbol{K}_{X,X} = \mathcal{K}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{N \times N} \\ \boldsymbol{K}_{X,\star} = \mathcal{K}(\boldsymbol{X}, \boldsymbol{X}_\star) \in \mathbb{R}^{N \times N_\star} \\ \boldsymbol{K}_{\star,\star} = \mathcal{K}(\boldsymbol{X}_\star, \boldsymbol{X}_\star) \in \mathbb{R}^{N_\star \times N_\star} \end{cases}$$

Prior Information

Prior Based on GP

In this case, the covariance of observed noisy responses is:

$$\begin{aligned}\mathrm{Cov}[y_i, y_j] =& \mathrm{Cov}[f(\boldsymbol{x}_i) + \epsilon_i, f(\boldsymbol{x}_j) + \epsilon_j] = \mathrm{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] + \mathrm{Cov}[\epsilon_i, \epsilon_j] \\ =& \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sigma_y^2 \delta_{ij}\end{aligned}$$

Thus we conclude:

$$\mathrm{Cov}[\boldsymbol{y}|\boldsymbol{X}] = \boldsymbol{K}_{X,X} + \sigma_y^2 \boldsymbol{I} \triangleq \boldsymbol{K}_\sigma$$

Based on GP, we have joint distribution $p(\boldsymbol{f}_X, \boldsymbol{f}_\star | \boldsymbol{X}, \boldsymbol{X}_\star)$ as:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_\star \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{x}} \\ \boldsymbol{\mu}_\star \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_\sigma & \boldsymbol{K}_{X,\star} \\ \boldsymbol{K}_{X,\star}^T & \boldsymbol{K}_{\star,\star} \end{bmatrix} \right)$$

Posterior By MVN Conditionals

Assuming we have observe $\boldsymbol{f}_X$, we can compute the posterior over $\boldsymbol{f}_\star$ using MVN conditionals.

Posterior for Noise Free Observations

MVN Conditionals

Suppose $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix}$ is jointly Gaussian with $\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}$, $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$ Then the posterior conditional is given by $p(\boldsymbol{y}_1|\boldsymbol{y}_2) = \mathcal{N}(\boldsymbol{y}_1|\boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2})$ where:

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{y}_2 - \boldsymbol{\mu}_2),\ \boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}$$

Calculating Posterior

Using prior distribution and MVN conditionals, we conclue $p(\boldsymbol{f}_\star|\mathcal{D}, \boldsymbol{X}_\star) = \mathcal{N}(\boldsymbol{f}_\star|\boldsymbol{\mu}_\star^{post}, \boldsymbol{\Sigma}_\star^{post})$ where:

$$\boldsymbol{\mu}_\star^{post} = \boldsymbol{\mu}_\star + \boldsymbol{K}_{X,\star}^T \boldsymbol{K}_\sigma^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_X)$$
$$\boldsymbol{\Sigma}_\star^{post} = \boldsymbol{K}_{\star,\star} - \boldsymbol{K}_{X,\star}^T \boldsymbol{K}_\sigma^{-1} \boldsymbol{K}_{X,\star}$$

Single Test Input

Suppose we use zero mean prior ($m(\boldsymbol{x}) = 0$) and kernel $\mathcal{K}(\cdot,\cdot)$. Assume the following case:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}, \ \boldsymbol{X}_\star = \begin{bmatrix} \boldsymbol{x}_\star^T \end{bmatrix}$$

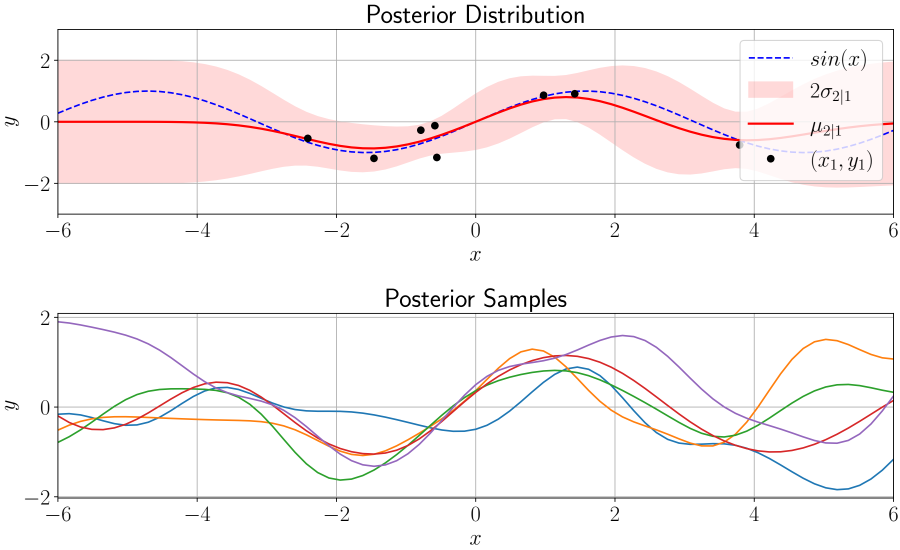
Then we can show that $p(f_\star|\mathcal{D}\boldsymbol{x}_\star) = \mathcal{N}(f_\star|0+\boldsymbol{k}_\star^T\boldsymbol{K}_\sigma^{-1}(\boldsymbol{y}-\boldsymbol{0}), k_{\star\star}-\boldsymbol{k}_\star^T\boldsymbol{K}_\sigma^{-1}\boldsymbol{k}_\star)$. where:

$$\boldsymbol{k}_\star = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}_\star,\boldsymbol{x}_1) \\ \vdots \\ \mathcal{K}(\boldsymbol{x}_\star,\boldsymbol{x}_N) \end{bmatrix}, \ \boldsymbol{k}_{\star\star} = \mathcal{K}(\boldsymbol{x}_\star,\boldsymbol{x}_\star)$$

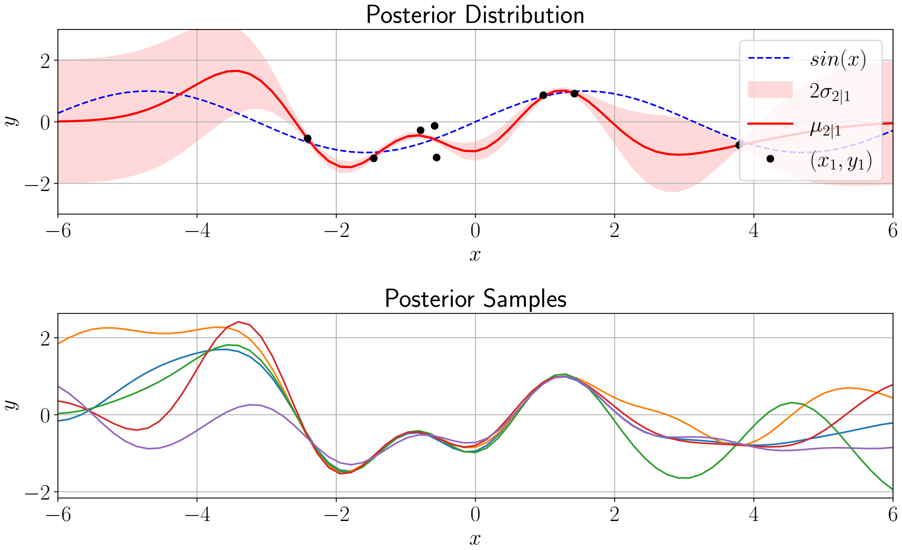
Thus we have conditional mean as:

$$\mu_{\star|X} = \boldsymbol{k}_\star^T(\boldsymbol{K}_\sigma^{-1}\boldsymbol{y}) \triangleq \boldsymbol{k}_\star^T\boldsymbol{\alpha} = \sum_{n=1}^{N}\mathcal{K}(\boldsymbol{x}_\star,\boldsymbol{x}_n)\alpha_n$$

Example: Posterior Distribution ($\sigma_y = 0.5$) [3]



Example: Posterior Distribution ($\sigma_y = 0.05$) [3]



Section 4

Support Vector Machines

Subsection 1

Hard Margin

Approach Definition

Support vector machines (SVMs) are non-probabilistic models for classification and regression formulated as:

$$f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_n)$$

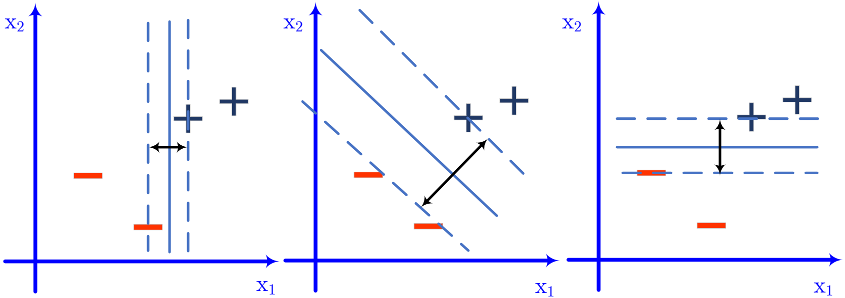
Idea Behind SVMs

- When N is large, kernel methods are not efficient.
- SVMs solve the aformentioned deficiency by ensuring that many of α_i coefficients are zero.
- *Support vectors* are training samples $\boldsymbol{x}_i$ whose corresponding coefficient α_i are not zero.

Widest Street Concept [4]

Hard Margin

In SVM with hard margin, we assume a two class classification problem where the training samples are linearly separable. We are looking for the widest classification margin.

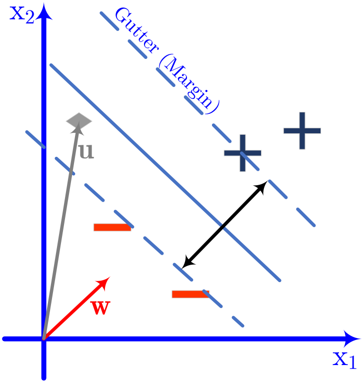


Decision Rule [4]

Intuition

Assume $\boldsymbol{w}$ to be perpendicular line to the decision street. As the projection of $\boldsymbol{u}$ on $\boldsymbol{w}$ increases, the unknown samples tends to lie on the + side. Thus:

$$\begin{cases} + & \text{if } \langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0 \geq 0 \\ - & \text{if } \langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0 < 0 \end{cases}$$



Toward Constraining

Considering a Margin

Assume samples $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ for positive and negative samples, respectively. Then we impose the following inequalities:

$$\langle \boldsymbol{w}, \boldsymbol{x}_+ \rangle + w_0 \geq 1$$
$$\langle \boldsymbol{w}, \boldsymbol{x}_- \rangle + w_0 \leq 1$$

We can encode the label using definition of y as:

$$y \triangleq \begin{cases} +1 & +\ samples \\ -1 & -\ samples \end{cases}$$

Using the definition of we can write the enequalities for positive and negative samples as:

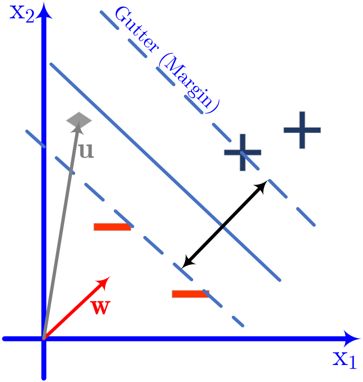
$$\left.\begin{array}{l} y_+ \left(\langle \boldsymbol{w}, \boldsymbol{x}_+ \rangle + w_0 \right) \geq 1 \\ y_- \left(\langle \boldsymbol{w}, \boldsymbol{x}_- \rangle + w_0 \right) \geq 1 \end{array}\right\} \Rightarrow y \left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0 \right) - 1 \geq 0$$

Samples on the Gutter

Samples on the Gutter

For samples on the gutter (margin), the inequality is reduced to equality as:

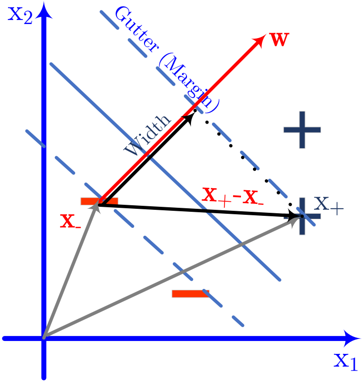
$$y\left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0\right) - 1 = 0$$



Width of the Street

The width of the street can be calculated uisng projection of difference vector $\boldsymbol{x}_+ = \boldsymbol{x}_-$ onto normalized normal vector $\boldsymbol{w}$ as:

$$\text{Width} = \left\langle \boldsymbol{x}_+ - \boldsymbol{x}_-, \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \right\rangle$$



Street Width

Equal Formulation

For positive samples on the gutter, we have:

$$y_+ \left(\langle \boldsymbol{w}, \boldsymbol{x}_+ \rangle + w_0 \right) = 1 \Rightarrow \langle \boldsymbol{w}, \boldsymbol{x}_+ \rangle = 1 - w_0$$

For negative samples on the gutter, we have:

$$y_- \left(\langle \boldsymbol{w}, \boldsymbol{x}_- \rangle + w_0 \right) = 1 \Rightarrow \langle \boldsymbol{w}, \boldsymbol{x}_- \rangle = -1 - w_0$$

Using the above equation, we conclude:

$$\begin{aligned} \text{Width} &= \left\langle \boldsymbol{x}_+ - \boldsymbol{x}_-, \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \right\rangle = \frac{1}{\|\boldsymbol{w}\|} \left(\langle \boldsymbol{x}_+, \boldsymbol{w} \rangle - \langle \boldsymbol{x}_-, \boldsymbol{w} \rangle \right) \\ &= \frac{1}{\|\boldsymbol{w}\|}(1 - w_0 + 1 + w_0) = \frac{2}{\|\boldsymbol{w}\|} \end{aligned}$$

Maximizing the Width

Optimization Problem

To maximize the width of street, we have to solve the following optimimzation problem:

$$\begin{aligned}\widehat{\boldsymbol{w}} &= \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{2}{\|\boldsymbol{w}\|} \text{ subject to } y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0\right) - 1 \geq 0, i = 1, \ldots, N \\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2 \text{ subject to } y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0\right) - 1 \geq 0, i = 1, \ldots, N\end{aligned}$$

Lagrangian

The Lagrangian for the above optimizaion problem is:

$$L = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0\right) - 1\right]$$

Differentiating the Lagrangian

With Respect to $\boldsymbol{w}$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x}_i = 0 \Rightarrow \widehat{\boldsymbol{w}} = \sum_i \alpha_i y_i \boldsymbol{x}_i$$

Thus $\boldsymbol{w}$ is linear combination of some of training samples $\boldsymbol{x}_i$

With Respect to w_0

$$\frac{\partial L}{\partial w_0} = -\sum_i \alpha_i y_i = 0 \Rightarrow \sum_i \alpha_i y_i = 0$$

Re-writing the Lagrangian

Re-writing the Lagrangian

Pulg-in the equation found by differentiating with respect to $\boldsymbol{w}$ and w_0 into Lagrangian, we have:

$$L = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\langle \boldsymbol{w}, \boldsymbol{w}_i \rangle + w_0 \right) - 1 \right]$$

$$= \frac{1}{2} \left\langle \sum_i \alpha_i y_i \boldsymbol{x}_i, \sum_j \alpha_j y_j \boldsymbol{x}_j \right\rangle - \sum_i \alpha_i y_i \left\langle \sum_j \alpha_j y_j \boldsymbol{x}_j, \boldsymbol{x}_i \right\rangle - w_0 \overbrace{\sum_i \alpha_i y_i}^{=0} + \sum_i \alpha_i$$

$$= \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle, \quad \sum_i \alpha_i y_i = 0$$

Lagrangian Form

As we can see, in the resulting Lagrangian, it's dependent upon input samples $\boldsymbol{x}_i$ is of the inner product form. Thus we just need to know the inner product of input samples for optimization.

Updating the Decision Rule

Updating the Decision Rule

Using equality $\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i$, we can write the decision rule as:

$$\begin{cases} +1 & \text{if } \sum_i \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + w_0 \geq 0 \\ -1 & \text{if } \sum_i \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + w_0 < 0 \end{cases}$$

Deciding on New Sample $\boldsymbol{x}$

As we can see, in the resulting decision rule, the dependency upon input query sample $\boldsymbol{x}$ and training input sample $\{\boldsymbol{x}_i\}_{i=1}^N$ is of the inner product form. Thus we just need to know the inner product of input samples and query sample to decide on the label.

Looking Back into Lagrangian

Dual Form

As we can see, the Lagrangian is:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle, \quad \sum_i \alpha_i y_i = 0$$

The above form is *dual form* of the objective function and the vector of Lagrange multipliers defined as $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$ can be found as:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \; \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \text{ subject to } \begin{cases} \sum_i \alpha_i y_i = 0 \\ \alpha_i \geq 0, i = 1, \ldots, N \end{cases}$$

Optimization Problem

The optimization problem for widest street can be formulated as:

$$= \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2 \text{ subject to } y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0\right) - 1 \geq 0, i = 1, \ldots, N$$

The above optimization problem is convex and we see the Lagrangian as:

$$L = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0\right) - 1\right]$$

Based on KKT conditions, we have:

$$\begin{aligned} &\alpha_i \geq 0,\ i = 1, \ldots, N \\ &y_i\left(\langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle + \widehat{w}_0\right) - 1 \geq 0 \\ &\alpha_i\left(y_i\left(\langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle + \widehat{w}_0\right) - 1\right) = 0 \end{aligned}$$

Support Vectors

Complementary Slackness

Based on *complementary slackness*, we have:

$$\begin{cases} y_i \left(\langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle + \widehat{w}_0 \right) - 1 = 0 \\ \text{or} \\ \alpha_i = 0 \end{cases}$$

Remember that the decision is made based on:

$$\begin{cases} +1 & \text{if } \sum_i \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + w_0 \geq 0 \\ -1 & \text{if } \sum_i \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + w_0 < 0 \end{cases}$$

Thus for each training input sample $\boldsymbol{x}_i$, either of the following condition may happend:

- $\alpha_i = 0$ and the sample is ignored in the decision.
- $y_i \left(\langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle + \widehat{w}_0 \right) = 1$ the sample is on the gutter.

Support Vectors

The input training samples lie on the gutter are known as support vectors. These samples contribute to the decision rule.

Calculating Parameters

As we see, the dual problem is defined as:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \text{ subject to } \begin{cases} \sum_i \alpha_i y_i = 0 \\ \alpha_i \geq 0, i = 1, \dots, N \end{cases}$$

Now consider the following definitions:

$$\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N \times 1}, \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \widehat{\boldsymbol{X}} = \operatorname{diag}(\boldsymbol{y}) \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}, \boldsymbol{S} = \widehat{\boldsymbol{X}} \widehat{\boldsymbol{X}}$$

Then we can rewrite the optimization as:

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha} \boldsymbol{S} \boldsymbol{\alpha} \text{ subject to } \begin{cases} \boldsymbol{y}^T \boldsymbol{\alpha} = 0 \\ \boldsymbol{\alpha} \geq 0 \end{cases}$$

The above problem can be solved using packages for standard quadratic programming (QP) solvers.

Calculating Parameters

Calculating Parameters

After calculating $\widehat{\alpha}$ using QP, we can find model parameters as:

- $\widehat{\boldsymbol{w}} = \sum_i \widehat{\alpha}_i y_i \boldsymbol{x}_i$
- $\widehat{w}_0$: Assume $\boldsymbol{x}_i$ to be a support vector, then we can calculate $\widehat{w}_0$ as:

$$y_i \left(\langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle + \widehat{w}_0 \right) = 1 \xrightarrow{\times y_i} \widehat{w}_0 = y_i - \langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle$$

In practice we average the value of $\widehat{w}_0$ resulting from all support vectors in set $\mathcal{S}$ as:

$$\widehat{w}_0 = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} (y_i - \langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(y_i - \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \boldsymbol{x}_j, \boldsymbol{x}_i \rangle \right)$$

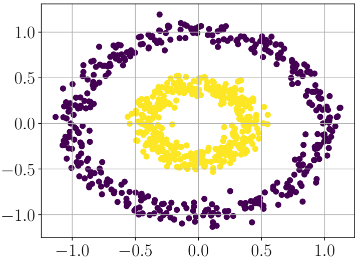
Subsection 2

Hard Margin SVM with Kernel

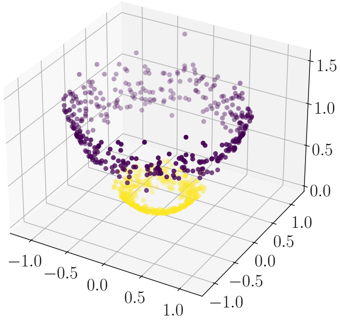
Linear Separability By Increasing Dimensions

Separability by Mapping

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{bmatrix} \Rightarrow \mathcal{K}\left(\overbrace{\begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}}^{\boldsymbol{x}_i}, \overbrace{\begin{bmatrix} x_{j1} \\ x_{j2} \end{bmatrix}}^{\boldsymbol{x}_j}\right) = x_{i1}x_{j1} + x_{i2}x_{j2} + (x_{i1}^2 + x_{i2}^2)(x_{j1}^2 + x_{j2}^2)$$



(a) Original dataset $\{(\boldsymbol{x}_i, y_i\}_{i=1}^N$



(b) Mapped dataset $\{(\boldsymbol{\phi}(\boldsymbol{x}_i), y_i\}_{i=1}^N$

A Closer Look to Hard Margin SVM

How to Solve Widest Margin Problem

Using QP, we solve:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \ \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}\boldsymbol{S}\boldsymbol{\alpha} \quad \text{subject to} \begin{cases} \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \\ \boldsymbol{\alpha} \geq 0 \end{cases}$$

where:

$$\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1}, \ \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}, \ \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \ \widehat{\boldsymbol{X}} = \operatorname{diag}(\boldsymbol{y}) \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}$$

$$\boldsymbol{S} = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}^T \Rightarrow \begin{cases} \boldsymbol{S} \in \mathbb{R}^{N\times N} \\ [\boldsymbol{S}]_{ij} = y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \end{cases}$$

To evaluate the class for a new data, we have:

$$\sum_i \widehat{\alpha}_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + \widehat{w}_0 \begin{matrix} \geq 0 & \Rightarrow y = +1 \\ < 0 & \Rightarrow y = -1 \end{matrix}$$

Replacing Inner Product with PD Kernel

Hard Margin SVM with Kernel

Assume using a mapping $\boldsymbol{\phi}: \mathbb{R}^D \to \mathcal{H}$, we convert dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ to $\{(\boldsymbol{\phi}(\boldsymbol{x}_i), y_i)\}_{i=1}^N$ where $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle$. Then we can design hard margin SVM as:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \; \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}\boldsymbol{S}\boldsymbol{\alpha} \quad \text{subject to } \begin{cases} \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \\ \boldsymbol{\alpha} \geq 0 \end{cases}$$

where:

$$\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1}, \; \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}, \; \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \; \widehat{\boldsymbol{X}} = \operatorname{diag}(\boldsymbol{y}) \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}$$

$$\boldsymbol{S} = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}^T \Rightarrow \begin{cases} \boldsymbol{S} \in \mathbb{R}^{N\times N} \\ [\boldsymbol{S}]_{ij} = y_i y_j \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle = y_i y_j \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{cases}$$

To evaluate the class for a new data, we have:

$$\sum_i \widehat{\alpha}_i y_i \underbrace{\langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}) \rangle}_{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)} + \widehat{w}_0 \begin{array}{ll} \geq 0 & \Rightarrow y = +1 \\ < 0 & \Rightarrow y = -1 \end{array}$$

Intuition

As we see before, for any positive kernel we have:

$$\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle$$

where $\boldsymbol{\phi} : \mathbb{R}^D \to \mathcal{H}$ and $\mathcal{H}$ is a Hilbert space. The process above can be describes as:

- Mapping input training samples $\{\boldsymbol{x}_i\}_{i=1}^N$ to Hilbert space $\mathcal{H}$ as $\{\boldsymbol{\phi}(\boldsymbol{x}_i)\}_{i=1}^N$
- Explicitly finding separating hyper-plane $\boldsymbol{w} = \sum_i \widehat{\alpha}_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i)$ (implicitly finding $\widehat{\boldsymbol{\alpha}}$)
- Finding $\widehat{w}_0$ as:

$$\widehat{w}_0 = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(y_i - \sum_{j \in \mathcal{S}} \alpha_j y_j \overbrace{\langle \boldsymbol{\phi}(\boldsymbol{x}_j), \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle}^{\mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_i)} \right)$$

- Evaluating new sample $\boldsymbol{x}$ as:

$$\sum_i \widehat{\alpha}_i y_i \underbrace{\langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}) \rangle}_{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)} + \widehat{w}_0 \begin{array}{ll} \geq 0 & \Rightarrow y = +1 \\ < 0 & \Rightarrow y = -1 \end{array}$$

Subsection 3

Soft Margin SVM

Soft Margin SVM

Not Linearly Separable Case

Assume the following case where the training dataset is not linearly separable. Then hard margin SVM cannot be solve due to the infeasibility of constraints.

SVM with Soft Margin

Formulation

Hard margin SVM is formulated as:

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 \ \text{ subject to } y_i\left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0\right) \geq 1$$

When the problem is not linearly separable, the set of vectors $\boldsymbol{w}$ and scalars w_0 that satisfy the constraints is empty.
In soft margin SVM, the above deficiency is solved by updating the problem as:

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_i \xi_i \ \text{ subject to } \begin{cases} y_i\left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

- $y_i\left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0\right) \geq 1 - \xi_i$ allows some samples to enter the margin or even cross the decision hyper-plane
- $C\sum_i \xi_i$ Controls the number of samples that enter the margin or cross the decision hyper-plane

Lagrangian

Lagrangian

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_i \xi_i \ \text{ subject to } \begin{cases} y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}\rangle + w_0\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

The Lagrangian for the above problem is:

$$\begin{aligned} L(\boldsymbol{w}, w_0, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}) =& \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_i \xi_i - \sum_i \alpha_i\left[y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + w_0\right) - 1 + \xi_i\right] \\ & - \sum_i \mu_i \xi_i \\ =& \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_i \xi - \sum_i \alpha_i y_i \langle \boldsymbol{w}, \boldsymbol{x}_i\rangle - \sum_i \alpha_i y_i w_0 \\ & + \sum_i \alpha_i - \sum_i \alpha_i \xi_i - \sum_i \mu_i \xi_i \end{aligned}$$

Lagrangian

Derivatives

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x}_i = 0 \Rightarrow \widehat{\boldsymbol{w}} = \sum_i \alpha_i y_i \boldsymbol{x}_i$$

$$\frac{\partial L}{\partial w_0} \sum_i \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0$$

Updating Langrangian Formulation

Using above equalities, we can rewrite Lagrangian in terms of dual variables ($\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$) as:

$$L = \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + C\sum_i \xi_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$
$$+ \sum_i \alpha_i - \sum_i \alpha_i \xi_i - \sum_i \mu_i \xi_i$$

On the other hand, we know:

$$C\sum_i \xi_i - \sum_i \alpha_i \xi_i - \sum_i \mu_i \xi_i = \sum_i (C - \alpha_i - \mu_i)\xi_i = \sum_i 0 \times \xi_i = 0$$

Dual Formulation

Dual Formulation

Thus we have the following optimization problem:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} -\frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_i \alpha_i \quad \text{subject to} \quad \begin{cases} 0 \leq \alpha_i \\ 0 \leq \mu_i \\ \sum_i \alpha_i y_i = 0 \\ C - \alpha_i - \mu_i = 0 \end{cases}$$

The effect of μ_i can be easily translated to upper bounding α_i as:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} -\frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_i \alpha_i \quad \text{subject to} \quad \begin{cases} 0 \leq \alpha_i \leq C \\ \sum_i \alpha_i y_i = 0 \end{cases}$$

Thus the only change in comparison to hard margin case, is the upper bound added to α_i.

Intuition

Intuition

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_i \xi_i \ \text{ subject to } \begin{cases} y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}\rangle + w_0\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

- For large values of C, the feasible interval for α_i is widened and we approach the hard margin SVM.
- For small values of C, the feasible interval for α_i becomes narrower and we allow more samples to cross the margin.

Slackness Complementary

Slackness Complementary

$$\alpha_i \left[y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0 \right) - 1 + \xi_i \right] = 0, \; i = 1, \dots, N$$
$$\mu_i \xi_i = 0, \; i = 1, \dots, N$$

Thus:

- $\alpha_i > 0 \Rightarrow \boldsymbol{x}_i$ is Suport Vector $\Rightarrow y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0 \right) = 1 - \xi_i$
  - $\mu_i > 0 \Rightarrow \begin{cases} \xi_i = 0 \Rightarrow \boldsymbol{x}_i \text{ is on the margin} \\ C - \alpha_i - \mu_i = 0 \Rightarrow \alpha_i < C \end{cases} \Rightarrow 0 < \alpha_i < C \rightarrow$ $\boldsymbol{x}_i$ is on the margin
  - $\xi_i > 0 \Rightarrow \begin{cases} \boldsymbol{x}_i \text{ crosses the margin} \\ \mu_i = 0 \Rightarrow \alpha_i = C \end{cases} \Rightarrow \alpha_i = C \rightarrow \boldsymbol{x}_i$ crosses the margin

Slackness Complementary

Slackness Complementary

$$\alpha_i \left[y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0 \right) - 1 + \xi_i \right] = 0, \ i = 1, \ldots, N$$
$$\mu_i \xi_i = 0, \ i = 1, \ldots, N$$

Thus:

- $y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0 \right) > 1 - \xi_i \Rightarrow \boldsymbol{x}_i$ is NOT Suport Vector $\Rightarrow \alpha_i = 0$

$$\left. \begin{array}{c} C - \alpha_i - \mu_i = 0 \\ \alpha_i = 0 \end{array} \right\} \Rightarrow \mu_i = C > 0 \Rightarrow \xi_i = 0 \Rightarrow y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0 \right) > 1$$

Therefore $\boldsymbol{x}_i$ is classified correctly and is not on the margin.

Calculating Parameters

Similar to hard SVM, the dual problem is defined as:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \; \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \text{ subject to } \begin{cases} \sum_i \alpha_i y_i = 0 \\ 0 \le \alpha_i \le C, i = 1, \dots, N \end{cases}$$

Now consider the following definitions:

$$\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1}, \; \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}, \; \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \; \widehat{\boldsymbol{X}} = \operatorname{diag}(\boldsymbol{y}) \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}, \; \boldsymbol{S} = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}$$

Then we can rewrite the optimization as:

$$\min_{\boldsymbol{\alpha}} \; \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{S}\boldsymbol{\alpha} \text{ subject to } \begin{cases} \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \\ 0 \le \boldsymbol{\alpha} \le C \end{cases}$$

The above problem can be solved using packages for standard quadratic programming (QP) solvers.

Calculating Parameters

Calculating Parameters

After calculating $\widehat{\alpha}$ using QP, we can find model parameters as:

- $\widehat{\boldsymbol{w}} = \sum_i \widehat{\alpha}_i y_i \boldsymbol{x}_i$
- $\widehat{w}_0$: Assume $\boldsymbol{x}_i$ to be a support vector *on the margin* $(0 < \alpha_i < C)$, then we can calculate $\widehat{w}_0$ as:

$$y_i \left(\langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle + \widehat{w}_0 \right) = 1 \xrightarrow{\times y_i} \widehat{w}_0 = y_i - \langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle$$

In practice we average the value of $\widehat{w}_0$ resulting from all support vectors on the margin in set $\mathcal{S}_m$ as:

$$\widehat{w}_0 = \frac{1}{|\mathcal{S}_m|} \sum_{i \in \mathcal{S}_m} \left(y_i - \langle \widehat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle \right) = \frac{1}{|\mathcal{S}_m|} \sum_{i \in \mathcal{S}_m} \left(y_i - \sum_{j \in \mathcal{S}_m} \alpha_j y_j \langle \boldsymbol{x}_j, \boldsymbol{x}_i \rangle \right)$$

Subsection 4

Soft Margin SVM with Kernel

A Closer Look to Soft Margin SVM

How to Solve Widest Margin Problem

Using QP, we solve:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\text{argmin}} \ \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}\boldsymbol{S}\boldsymbol{\alpha} \quad \text{subject to} \ \begin{cases} \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \\ 0 \leq \boldsymbol{\alpha} \leq C \end{cases}$$

where:

$$\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1}, \ \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}, \ \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \ \widehat{\boldsymbol{X}} = \text{diag}(\boldsymbol{y}) \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}$$

$$\boldsymbol{S} = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}^T \Rightarrow \begin{cases} \boldsymbol{S} \in \mathbb{R}^{N\times N} \\ [\boldsymbol{S}]_{ij} = y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \end{cases}$$

To evaluate the class for a new data, we have:

$$\sum_i \widehat{\alpha}_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + \widehat{w}_0 \ \begin{matrix} \geq 0 & \Rightarrow y = +1 \\ < 0 & \Rightarrow y = -1 \end{matrix}$$

Replacing Inner Product with PD Kernel

Soft Margin SVM with Kernel

Assume using a mapping $\boldsymbol{\phi}: \mathbb{R}^D \to \mathcal{H}$, we convert dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ to $\{(\boldsymbol{\phi}(\boldsymbol{x}_i), y_i)\}_{i=1}^N$ where $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle$. Then we can design soft margin SVM as:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \; \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha} \boldsymbol{S} \boldsymbol{\alpha} \quad \text{subject to} \begin{cases} \boldsymbol{y}^T \boldsymbol{\alpha} = 0 \\ 0 \leq \boldsymbol{\alpha} \leq C \end{cases}$$

where:

$$\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N \times 1}, \; \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}, \; \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \; \widehat{\boldsymbol{X}} = \operatorname{diag}(\boldsymbol{y}) \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}$$

$$\boldsymbol{S} = \widehat{\boldsymbol{X}} \widehat{\boldsymbol{X}}^T \Rightarrow \begin{cases} \boldsymbol{S} \in \mathbb{R}^{N \times N} \\ [\boldsymbol{S}]_{ij} = y_i y_j \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle = y_i y_j \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{cases}$$

To evaluate the class for a new data, we have:

$$\sum_i \widehat{\alpha}_i y_i \underbrace{\langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}) \rangle}_{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)} + \widehat{w}_0 \begin{array}{ll} \geq 0 & \Rightarrow y = +1 \\ < 0 & \Rightarrow y = -1 \end{array}$$

Intuition

Intuition

As we see before, for any positive kernel we have:

$$\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle$$

where $\boldsymbol{\phi} : \mathbb{R}^D \to \mathcal{H}$ and $\mathcal{H}$ is a Hilbert space. The process above can be describes as:

- Mapping input training samples $\{\boldsymbol{x}_i\}_{i=1}^N$ to Hilbert space $\mathcal{H}$ as $\{\boldsymbol{\phi}(\boldsymbol{x}_i)\}_{i=1}^N$
- Explicitly finding separating hyper-plane $\boldsymbol{w} = \sum_i \widehat{\alpha}_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i)$ (implicitly finding $\widehat{\boldsymbol{\alpha}}$)
- Finding $\widehat{w}_0$ as:

$$\widehat{w}_0 = \frac{1}{|\mathcal{S}_m|} \sum_{i \in \mathcal{S}_m} \left(y_i - \sum_{j \in \mathcal{S}_m} \alpha_j y_j \overbrace{\langle \boldsymbol{\phi}(\boldsymbol{x}_j), \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle}^{\mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_i)} \right)$$

- Evaluating new sample $\boldsymbol{x}$ as:

$$\sum_i \widehat{\alpha}_i y_i \underbrace{\langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}) \rangle}_{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)} + \widehat{w}_0 \begin{array}{ll} \geq 0 & \Rightarrow y = +1 \\ < 0 & \Rightarrow y = -1 \end{array}$$

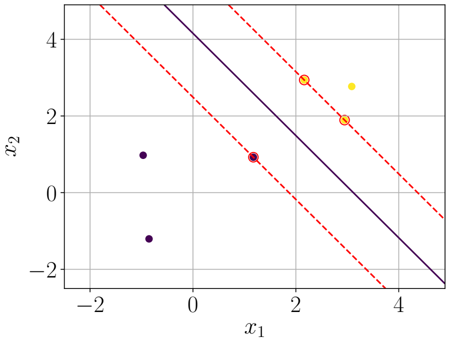
Subsection 5

Samples of SVM

Hard Margin SVM

Hard Margin SVM

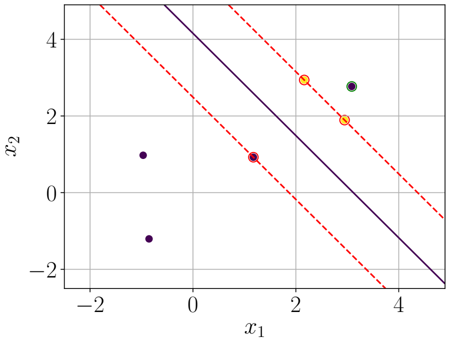
$$\boldsymbol{X} = \begin{bmatrix} 2.16 & 2.94 \\ 2.95 & 1.89 \\ 3.09 & 2.77 \\ 1.17 & 0.92 \\ -0.97 & 0.98 \\ -0.85 & -1.21 \end{bmatrix}, \ \boldsymbol{y} = \begin{bmatrix} 1.00 \\ 1.00 \\ 1.00 \\ -1.00 \\ -1.00 \\ -1.00 \end{bmatrix}, \ \Rightarrow \widehat{\alpha} = \begin{bmatrix} 0.1105 \\ 0.3898 \\ 0.0000 \\ 0.5003 \\ 0.0000 \\ 0.0000 \end{bmatrix}, \ \widehat{w}_0 = \begin{bmatrix} -2.4943 \end{bmatrix}$$



Soft Margin SVM

Soft Margin SVM with $C = 10$

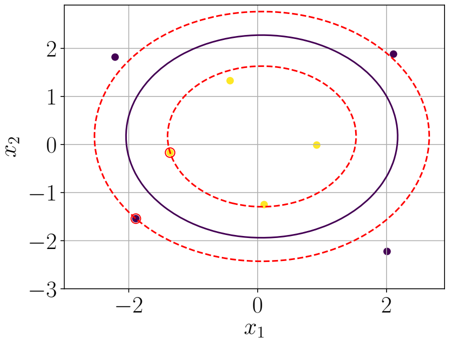
$$\boldsymbol{X} = \begin{bmatrix} 2.16 & 2.94 \\ 2.95 & 1.89 \\ 3.09 & 2.77 \\ 1.17 & 0.92 \\ -0.97 & 0.98 \\ -0.85 & -1.21 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} 1.00 \\ 1.00 \\ -1.00 \\ -1.00 \\ -1.00 \\ -1.00 \end{bmatrix}, \quad \Rightarrow \widehat{\alpha} = \begin{bmatrix} 5.5408 \\ 8.1495 \\ 10.0000 \\ 3.6904 \\ 0.0000 \\ 0.0000 \end{bmatrix}, \quad \widehat{w}_0 = \begin{bmatrix} -2.4943 \end{bmatrix}$$



Hard Margin kernel-SVM

Hard Margin kernel-SVM

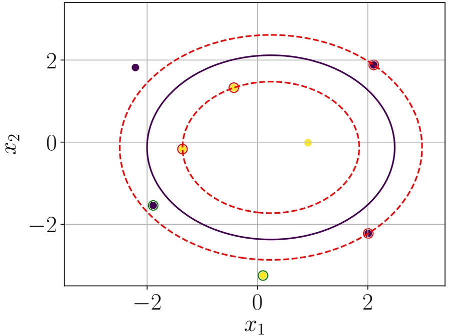
$$\boldsymbol{X} = \begin{bmatrix} 0.92 & -0.01 \\ -0.43 & 1.33 \\ -1.36 & -0.17 \\ 0.10 & -1.25 \\ -2.21 & 1.82 \\ -1.89 & -1.54 \\ 2.01 & -2.22 \\ 2.11 & 1.88 \end{bmatrix}, \ \boldsymbol{y} = \begin{bmatrix} 1.00 \\ 1.00 \\ 1.00 \\ 1.00 \\ -1.00 \\ -1.00 \\ -1.00 \\ -1.00 \end{bmatrix}, \ \Rightarrow \widehat{\alpha} = \begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.1066 \\ 0.0000 \\ 0.0000 \\ 0.1066 \\ 0.0000 \\ 0.0000 \end{bmatrix}, \ \widehat{w}_0 = \begin{bmatrix} 1.9156 \end{bmatrix}$$



Soft Margin kernel-SVM

Soft Margin kernel-SVM with $C = 10$

$$\boldsymbol{X} = \begin{bmatrix} 0.92 & -0.01 \\ -0.43 & 1.33 \\ -1.36 & -0.17 \\ 0.10 & -3.25 \\ -2.21 & 1.82 \\ -1.89 & -1.54 \\ 2.01 & -2.22 \\ 2.11 & 1.88 \end{bmatrix}, \ \boldsymbol{y} = \begin{bmatrix} 1.00 \\ 1.00 \\ 1.00 \\ 1.00 \\ -1.00 \\ -1.00 \\ -1.00 \\ -1.00 \end{bmatrix}, \ \Rightarrow \hat{\alpha} = \begin{bmatrix} 0.0000 \\ 2.7729 \\ 3.8414 \\ 10.0000 \\ 0.0000 \\ 10.0000 \\ 6.4263 \\ 0.1879 \end{bmatrix}, \ \hat{w}_0 = \begin{bmatrix} 2.0089 \end{bmatrix}$$



Subsection 6

Multiclass SVM

One-Versus-Rest Strategy

Approach

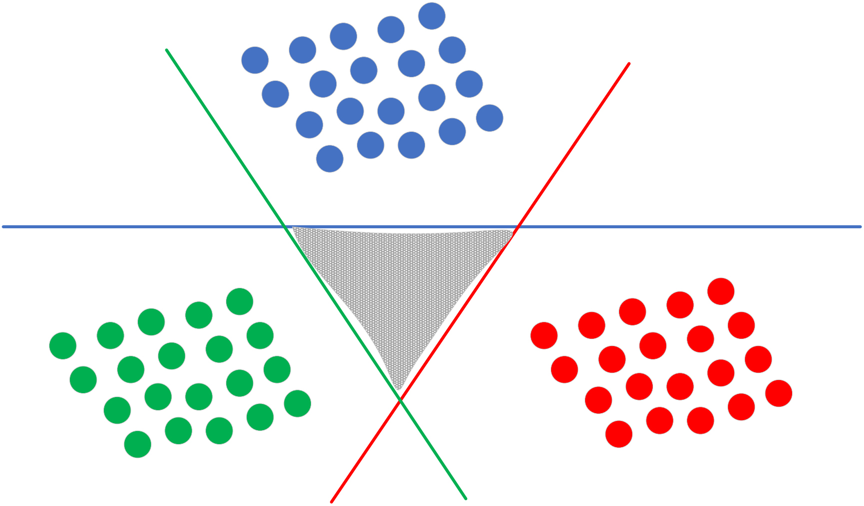
Assume we have C classes labels as $y \in \{1, 2, \ldots, C\}$. Then:

- For each value of $c \in \{1, \ldots, C\}$, we train a SVM ($f_c(\boldsymbol{x}) = \boldsymbol{W}^{(c)}\boldsymbol{x} + w_0^{(c)}$) over the following dataset:
  - Samples belonging to class c are labeled as $+1$
  - Samples belonging to classes other than c are labeled as -1
- For a new sample $\boldsymbol{x}$, we find the label as: $\widehat{y}(\boldsymbol{x}) = \text{argmax}_c\, f_c(\boldsymbol{x})$

Disadvantages

- Some regions in this method becomes ambiguous (Regions where $f_c(\boldsymbol{x}) < 0, c = 1, \ldots, C$)
- The value of $f_c(\boldsymbol{x})$ are not calibrated (having $f_1(\boldsymbol{x}) = l$ and $f_2(\boldsymbol{x}) = l$ is not informative due to un-calibrated function values)
- If the original dataset is balances across different classes, then the dataset generated to train each $f_i(\boldsymbol{x})$ is imbalanced.

One-Versus-Rest Strategy



One-Versus-One Strategy

Approach

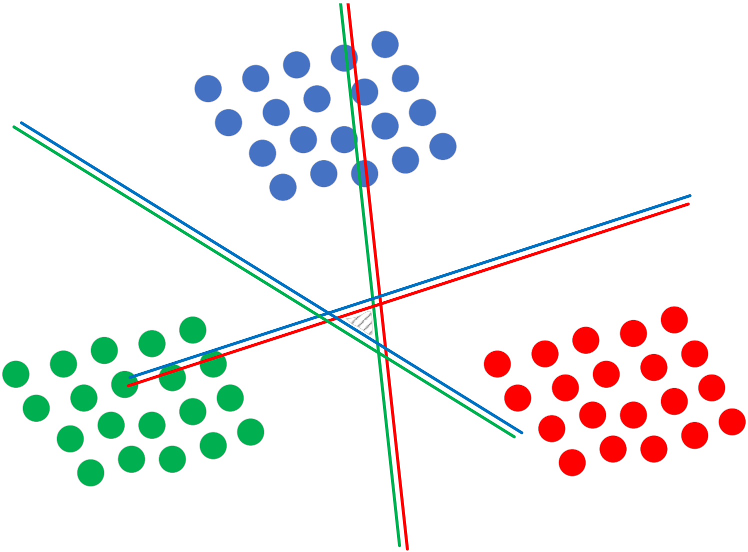
Assume we have C classes labels as $y \in \{1, 2, \ldots, C\}$. Then:

- For each possible pair of labels $(c, k) : c, k \in \{1, \ldots, C\}$, we train a SVM $(f_{ck}(\boldsymbol{x}) = \boldsymbol{W}^{(ck)}\boldsymbol{x} + w_0^{(ck)})$ over the following dataset:
  - Samples belonging to class c are labeled as $+1$
  - Samples belonging to class k are labeled as -1
- For a new sample $\boldsymbol{x}$, we find the label as: $\widehat{y}(\boldsymbol{x}) = \text{MaxVote}(\{f_{ck}(\boldsymbol{x}\}))$

Disadvantages

- Some regions in this method becomes ambiguous (Equally voted regions)
- We need to train $\mathcal{O}(C^2)$ models

One-Versus-Rest Strategy



References I

“Kernels and kernel methods (10/09/13),” https://www.cs.princeton.edu/~bee/courses/scribe/lec_10_09_2013.pdf, Accessed: 2022-09-24.

“Gaussian processes (3/3) - exploring kernels,” https://peterroelants.github.io/posts/gaussian-process-kernels/, Accessed: 2022-09-24.

“Gaussian processes (1/3) - from scratch,” https://peterroelants.github.io/posts/gaussian-process-tutorial/, Accessed: 2022-09-24.

“16. learning: Support vector machines,” https://www.youtube.com/watch?v=_PwhiWxHK8o&t=1604s, Accessed: 2022-09-24.