CS:E4830 Kernel Methods in Machine Learning

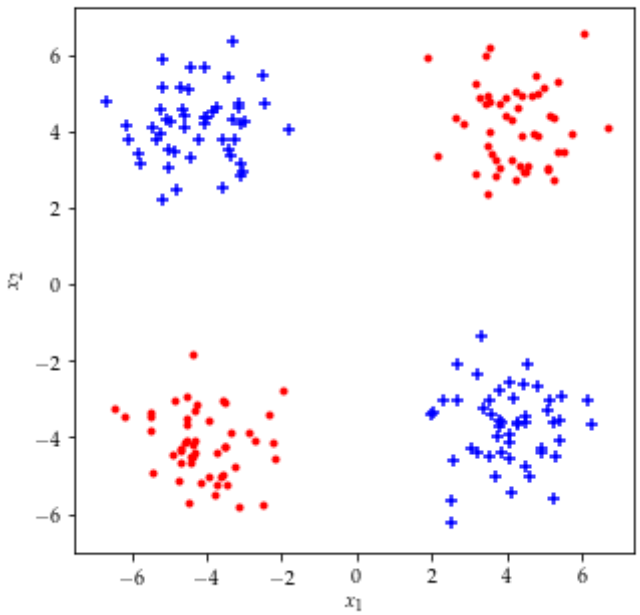
Lecture 10 : Course Review

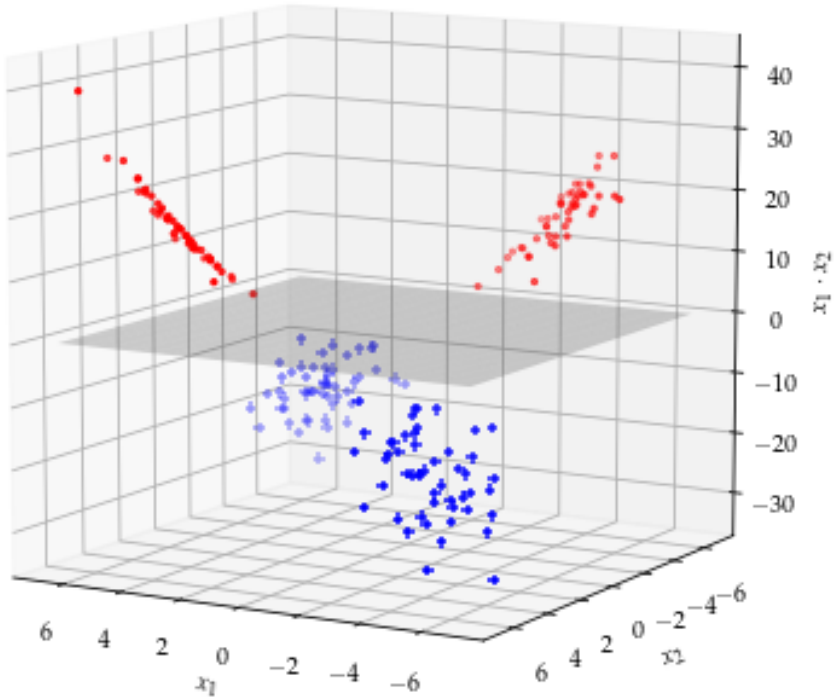
Rohit Babbar

12th May, 2021

Explicit Feature Mapping

- Dataset in 2-D (left), which is not linearly separable can be separated by a plane in 3-D (third feature is the product $x_1 x_2$)





Kernel Methods - Motivation

- Most learning algorithms such as Support Vector Machines, and Logistic regression (classification part used in deep networks) can be written in the form of Inner/dot product between vectors in the feature space $\phi(x_i)$s, i.e. $\langle \phi(x_i), \phi(x_j) \rangle$.
- The prediction function has the following form :

$$f(x) = \text{Some function of} \left(\sum_{i=1}^{N} \langle \phi(x_i), \phi(x) \rangle \right)$$

- Kernels are functions which give us the dot product $\langle \phi(x_i), \phi(x_j) \rangle$ directly without explicitly computing the feature expansion $\phi(.)$

Properties of Kernels

- **Positive Scalar Multiple** - For any $\alpha > 0$, if $k(.,.)$ is a kernel, then $\alpha k(.,.)$ is also a kernel.
- **Conic Sum of Kernels** - For kernels $(k_j)_{j=1}^K$, and $(\alpha_j)_{j=1}^K > 0$, $\sum_{j=1}^K \alpha_j k_j$ is also a kernel
- **Difference of Kernels** is not necessarily a kernel
- **Product** - product of kernels is a also kernel
- **Mappings** - For an arbitrary function $f : \mathcal{X} \mapsto \mathbb{R}$, and a kernel $k(.,.)$, $\hat{k}(x, x') = f(x)k(x, x')f(x')$ is also a kernel
- Used above properties to prove that polynomial, exponential, and Gaussian kernels are valid kernel functions.

Positive Definite Functions

Definition - Positive definite functions

A symmetric function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is positive definite if $\forall N \geq 1, \forall (a_1, \ldots, a_N) \in \mathbb{R}^N, \forall (x_1, \ldots, x_N) \in \mathcal{X}^N$,

$$\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j k(x_i, x_j) \geq 0$$

Moore-Aronszajn Theorem

A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a kernel if and only if it is symmetric and positive definite.

The kernel matrix

- A **kernel matrix** (also called the **Gram matrix**), is an $N \times N$ matrix of pairwise similarity values is used:

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

- Each entry is an inner product between two data points $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, where $\phi(.)$ is a feature map in vector form
- Since an inner product is symmetric, therefore K is a symmetric matrix
- In addition, K is positive definite

RKHS - Definition I

Definition (RKHS)

Let $\mathcal{H}$ be a Hilbert space of real-valued **functions** on the input $\mathcal{X}$. Then $\mathcal{H}(\subset \mathcal{R}^{\mathcal{X}})$ is defined to an **Reproducing kernel Hilbert Space (RKHS)** with $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ as the reproducing kernel, if the following conditions are satisfied

- $\forall x \in \mathcal{X}, k(., x) \in \mathcal{H}$ i.e., the space $\mathcal{H}$ contains all functions of the form $k(., x)$ for every element x in the input space $\mathcal{X}$,
- $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}$, the following property holds :$f(x) = \langle f, k(., x) \rangle_{\mathcal{H}}$ (it is called the reproducing property of the kernel).

RKHS - Definition II

Definition (RKHS)

Let $\mathcal{H}$ be a Hilbert space of real-valued **functions** on the input $\mathcal{X}$. Then $\mathcal{H}(\subset \mathcal{R}^{\mathcal{X}})$ is defined to be an **Reproducing kernel Hilbert Space (RKHS)** if and only if, for any element x in the input space $\mathcal{X}$, the following function F_x, which takes a function f from the Hilbert Space $\mathcal{H}$, and maps it to its value $f(x) \in \mathbb{R}$

$$\begin{array}{rccc} F_x: & \mathcal{H} & \mapsto & \mathbb{R} \\ & f & \mapsto & f(x) \end{array}$$

is continuous

We saw the equivalence between these two definitions

RKHS norm controls smoothness

RKHS norm and smoothness

$$
\begin{aligned}
|f(x) - f(x')| &= |\langle f, k(x,.)\rangle - \langle f, k(x',.)\rangle| \quad \text{(reproducing property applied to } f\text{)} \\
&= |\langle f, k(x,.) - k(x',.)\rangle| \quad \text{(linearity of dot product)} \\
&\leq ||k(.,x) - k(.,x')||_{\mathcal{H}} ||f||_{\mathcal{H}} \quad \text{(by Cauchy-Schwarz inequality)}
\end{aligned}
$$

- $||f||_{\mathcal{H}}$ controls how much the values at two points x and x' differ compared to their distance
- Larger value of $||f||_{\mathcal{H}}$ allows higher variations (potentially non-smooth functions)

Smaller RKHS norm $\implies$ Smooth functions

- The same happens in finite diemsions when we add regularization $||w||^2$ for linear regression and SVM

Notion of Generalization

It is desired that the error of our classifier is close to that of Bayes classifier. However, another desirable quality in machine learning algorithms is **Generalization**

- Let f_n be a classifier obtained by some algorithm (such as deep net or SVM or Random forest) which is based on a finite training sample of size n.
- The classifier f_n generalizes well if the difference between empirical and expected of f_n is low, i.e.,

$$|R(f_n) - R_{emp}(f_n)| \approx 0$$

- Note that having low generalization gap does imply low expected or test error, it just means that **empirical error is a good indicator of expected error**

Large vs Small Function class

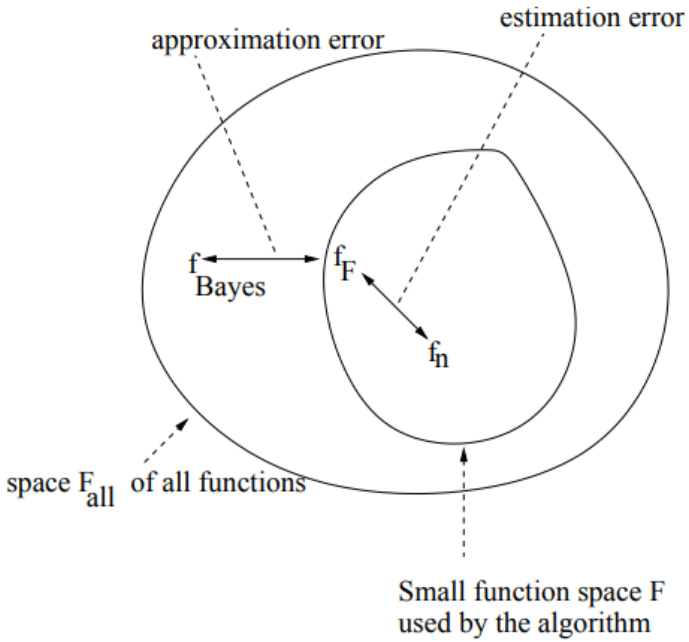


Figure: Pictorial depiction of the components of classification error

- The space F_{all} contains all possible functions that may be implmented using SVM, Deep nets, Random Forest and everything else
- **Estimation error** - $(R(f_n) - R(f_{\mathcal{F}}))$ - **finiteness of training data**
- **Approximation error** - $(R(f_{\mathcal{F}}) - R(f_{Bayes}))$ - **choice of function class**
- For example - If someone is claiming that using a deep net on a certain ML problem works better than SVM, which of the two errors is actually going down?

Large vs Small Function class

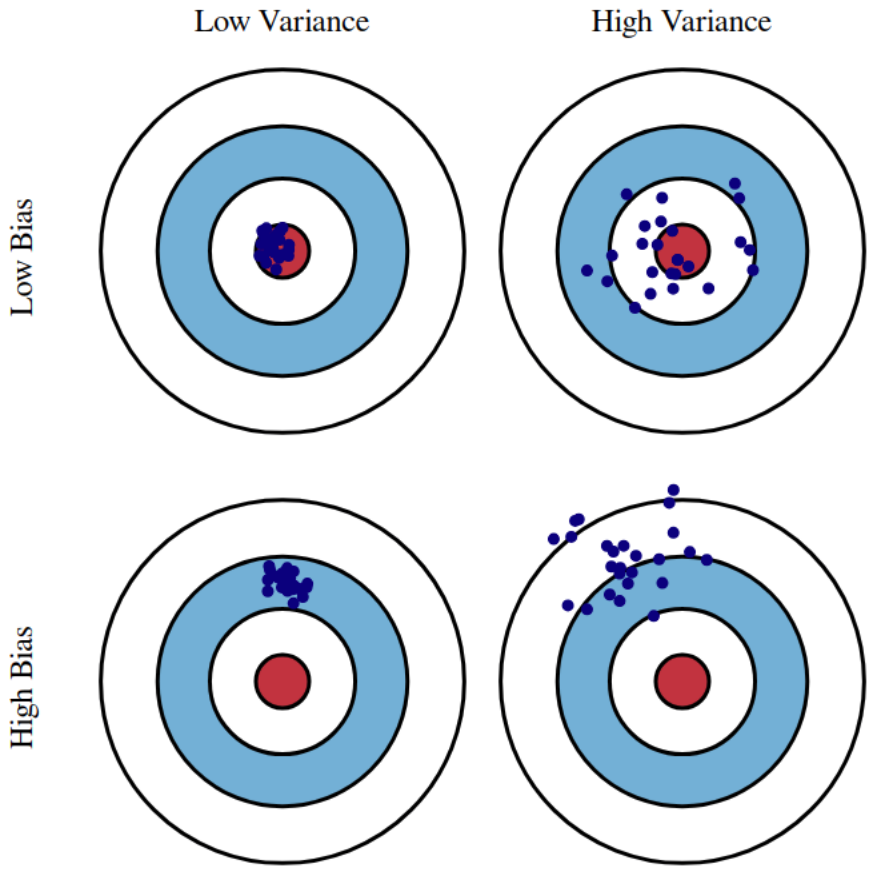


Figure: Pictorial depiction of the components of classification error

- **Estimation error** - $(R(f_n) - R(f_{\mathcal{F}}))$ - corresponds to **Variance**
- **Approximation error** - $(R(f_{\mathcal{F}}) - R(f_{Bayes}))$ - corresponds to **Bias**

Error variation with Function class capacity

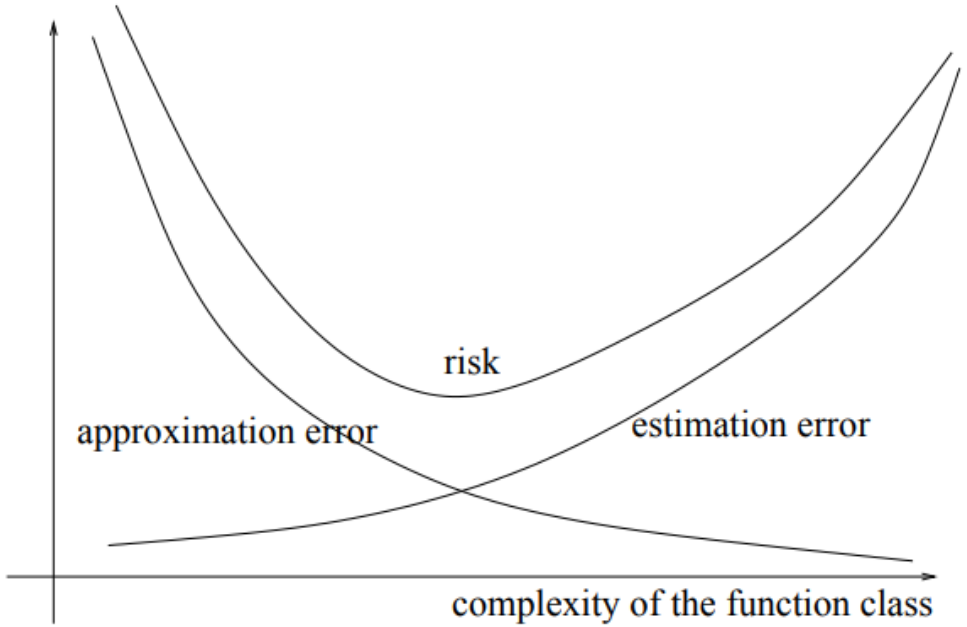


Figure: Variation of error components with the complexity of function class (tutorial by Von Luxburg and Schoelkopf)

- To the left with low complexity function class -
  - Linear classifiers or kernel classifier with high variance
- To the right with high complexity function class -
  - Deep neural networks

Empirical Risk Minimization

In practice, learning algorithms (do not have access to the underlying data generating distribution P over $\mathcal{X} \times \mathcal{Y}$) are based on minimizing error on the training data. Formally, this is given as follows :

Principle of ERM

The idea behind the principle of Empirical Risk Minimization is to find a classifier in a pre-defined function class which minimizes the empirical risk. That is

$$f_n := \arg\min_{f \in \mathcal{F}} R_{emp}(f)$$

- We want to check if the classifier (function) f_n that we learn from ERM is consistent or not

$$P(R(f_n) - R(f_{\mathcal{F}}) > \epsilon) \to 0 \text{ as } n \to \infty$$

Uniform Convergence

- **Uniform Convergence** is a condition over a function class which ensures consistency of ERM, and is given by $|R_{emp}(f) - R(f)| < \epsilon, \forall f \in \mathcal{F}$ for some finite sample size n
- Alternatively, the condition of Uniform Convergence can be stated $\sup_{f \in \mathcal{F}} |R_{emp}(f) - R(f)| < \epsilon$

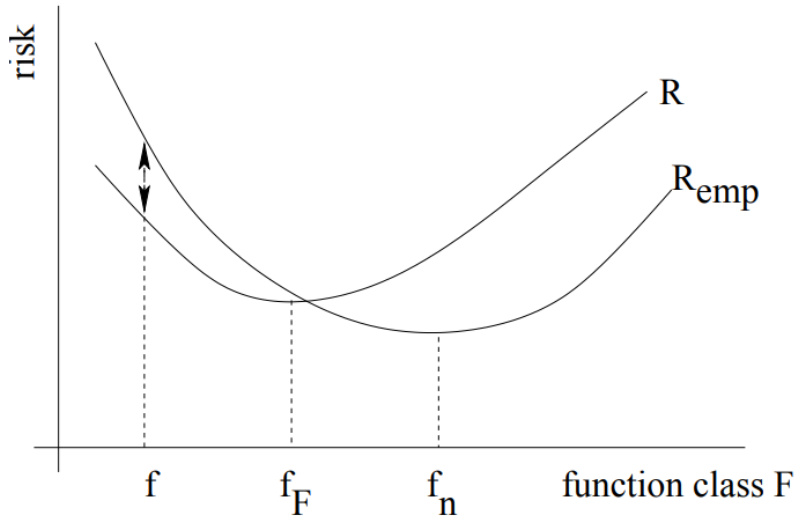


Figure: Under Uniform Convergence, the difference between the two curves becomes arbitrarily small for some large but finite sample size n

NASC for consistency of ERM

- Uniform convergence is a sufficient condition for the consistency of ERM
- Is it also necessary?

Theorem by Vapnik and Chervonenkis

Uniform convergence, i.e.,

$$\mathbb{P}(\sup_{f \in \mathcal{F}} |R(f) - R_{emp}(f)| > \epsilon) \to 0 \text{ as } n \to \infty$$

$\forall \epsilon > 0$ is a necessary and sufficient condition for consistency of ERM with respect to the function class $\mathcal{F}$.

Capacity of Function Class

The main quantity of interest from the previous theorem is the following :

$$\mathbb{P}(\sup_{f \in \mathcal{F}} |R(f) - R_{emp}(f)| > \epsilon)$$

- Can we study the above quantity in the non-asymptotic regime, i.e. when the sample size n is finite
  - Practically, this also matters more since we normally have finite data size
- In bounding the quantity $\mathbb{P}(\sup_{f \in \mathcal{F}} |R(f) - R_{emp}(f)| \geq \epsilon)$, there are two challenges :
  - Infinitely many functions, due to **continuous nature of the function class**
  - The expected risk $R(f)$, which depends on the underlying probability distribution, and **cannot be computed from training data**
- To get a handle on this, we need the following three concepts :
  - Union bound - $\mathbb{P}(\sup_{f \in \mathcal{F}} |R(f) - R_{emp}(f)| \geq \epsilon) \leq 2m \exp(-2n\epsilon^2)$
  - Symmetrization - $\mathbb{P}(\sup_{f \in \mathcal{F}} |R(f) - R_{emp}(f)| > \epsilon) \leq 2\mathbb{P}(\sup_{f \in \mathcal{F}} |R_{emp}(f) - R'_{emp}(f)| > \epsilon/2)$
  - Shattering - $\mathbb{P}(\sup_{f \in \mathcal{F}} |R(f) - R_{emp}(f)| \geq \epsilon) \leq 2\mathcal{N}(\mathcal{F}, 2n) \exp(-n\epsilon^2/4)$

Representer Theorem

- For the following optimization

$$f_{\mathcal{H}} := \arg\min_{f} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda\theta(||f||_{\mathcal{H}})$$

  where $\theta : [0, \infty) \mapsto \mathbb{R}$ is non-decreasing function
- Even though the above problem is potentially an infinite dimensional optimization problem, **Representer Theorem** states its solution can be expressed in the following form

$$f_{\mathcal{H}} = \sum_{i=1}^{N} \alpha_i k(., x_i)$$

  where $\alpha_i \in \mathbb{R}$
- Infinite to finite dimensional problem

Solving Kernel Ridge Regression

- Let's denote by
  - $y \in \mathbb{R}^N$, the label vector denoting the true values for the inputs
  - The kernel matrix K, where $K_{ij} = K(x_i, x_j)$
  - $\boldsymbol{\alpha} \in \mathbb{R}^N$, the co-efficients we want to find
- For the input instance, the prediction by the desired function can be written as follows :

$$(\hat{f}(x_1), \ldots, \hat{f}(x_N))^T = K\boldsymbol{\alpha}$$

- We also know that

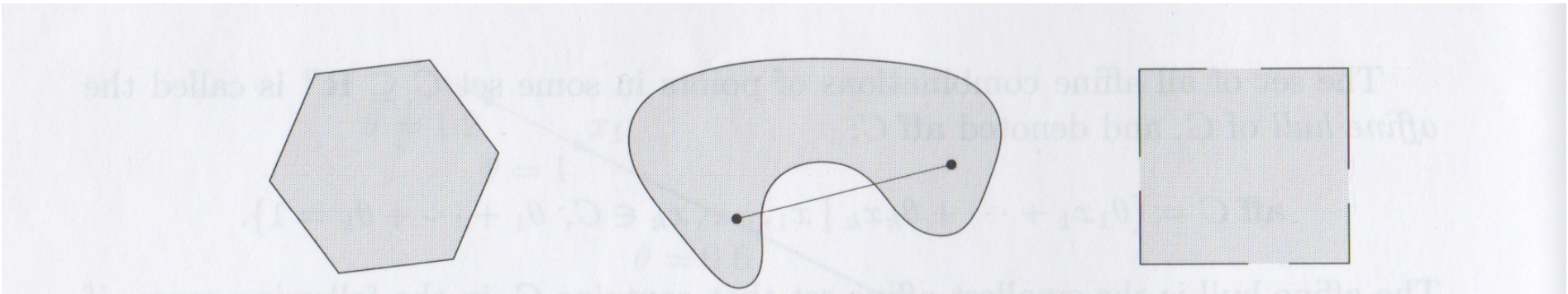
$$||f||^2_{\mathcal{H}} = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

- Solving Kernel Ridge Regression involves solving

$$\arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \frac{1}{N}(K\boldsymbol{\alpha} - y)^T(K\boldsymbol{\alpha} - y) + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

Convex sets

- A **line segment** between $x_1 \in \mathbb{R}^d$ and $x_2 \in \mathbb{R}^d$ is defined as all points that satisfy
$$x = \theta x_1 + (1 - \theta)x_2, 0 \leq \theta \leq 1$$
- A **convex set** contains the line segment between any two distinct points in the set
$$x_1, x_2 \in C, 0 \leq \theta \leq 1 \Rightarrow \theta x_1 + (1 - \theta)x_2 \in C$$
- Below: Convex and non-convex sets. Q: Which ones are convex?

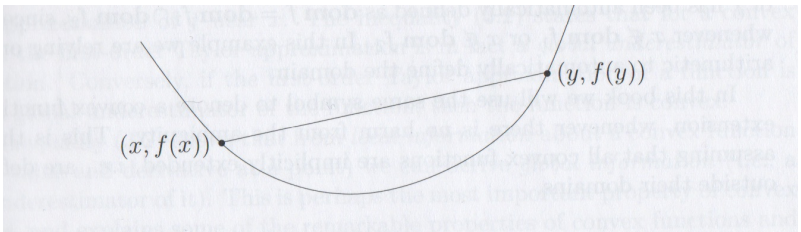


Convex functions

- A function $f : \mathbb{R}^d \mapsto \mathbb{R}$ is convex if (i) the domain of f is a convex set and (ii) for all x, y, and $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y).$$

- Geometrical interpretation: the graph of the function lies below the line segment from $(x, f(x))$ to $(y, f(y))$



- A function f is
  - strictly convex if strict inequality holds above
  - concave if $-f$ is convex.

Duality: Lagrangian

- Consider the primal optimisation problem

$$\begin{aligned} \min_{x \in \mathcal{D}} \ & f_0(x) \\ \text{s.t.} \ & f_i(x) \leq 0, i = 1, \ldots, m \\ & h_i(x) = 0, i = 1, \ldots, p \end{aligned}$$

  with variable $x \in \mathbb{R}^d$

- Augment the objective function with the weighted sum of the constraint functions to form the **Lagrangian** of the optimization problem:

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

- $\lambda_i, i = 1, \ldots, m$ and $\nu_i, i = 1, \ldots, p$ (ν is the greek letter 'nu') are called the **Lagrange multipliers** or **dual variables**

Lagrange dual function

- The **Lagrange dual function** $g : \mathbb{R}^m \times \mathbb{R}^p \mapsto \mathbb{R}$ is the minimum value of the Lagrangian over x:

$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) = \inf_x \{f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)\}$$

- Intuitively:
  - Fixing coefficients (λ, ν) corresponds to certain level of penalty,
  - The infimum returns the optimal x for that level of penalty
  - $g(\lambda, \nu)$ is the corresponding value for the Lagrangian
  - $g(\lambda, \nu)$ is a concave function as a pointwise infimum of a family of affine functions of (λ, ν)

The Lagrange dual problem

- For each pair $(\lambda, \nu), \lambda \geq 0$, the Lagrange dual function gives a lower bound on the optimal value of p^*.
- What is the tightest lower bound that can be achieved? We need to find the maximum
- This gives us a optimization problem

$$\max_{\lambda,\nu} g(\lambda, \nu)$$
$$\text{s.t.} \lambda \geq 0$$

- It is called the **Lagrange dual problem** of the original optimization problem.
- It is a convex optimisation problem, since it is equivalent to minimising $-g(\lambda, \nu)$ which is a convex function

SVM Problem Formulation

- Using Representer theorem, the problem can be reformulated as

$$\min_{\alpha \in \mathbb{R}^N} \left\{ \frac{1}{N} \sum_{i=1}^{N} \ell_{hinge}(y_i [K\alpha]_i) + \lambda \alpha^T K \alpha \right\}$$

- The above optimization problem is convex
- However, it is non-smooth optimization problem

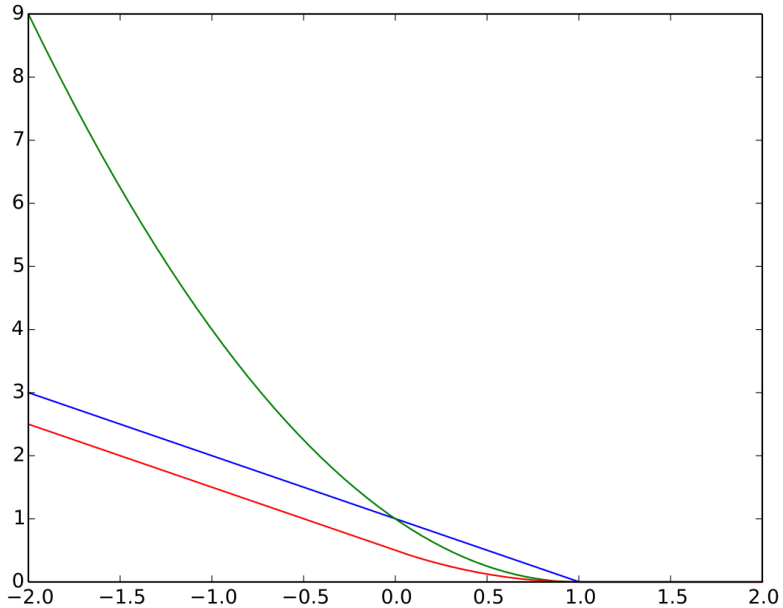


Figure: $z = yf(x)$ in the above graph

Rewriting in terms of Primal Variables

SVM Primal Formulation

$$\min_{\alpha \in \mathbb{R}^N, \xi \in \mathbb{R}^N} \left\{ \frac{1}{N} \sum_{i=1}^{N} \xi_i + \lambda \alpha^T K \alpha \right\}$$

such that

$$\begin{cases} 1 - y_i [K\alpha]_i - \xi_i \leq 0 & \text{for } i = 1, \ldots, N \\ -\xi_i \leq 0 & \text{for } i = 1, \ldots, N \end{cases}$$

SVM Dual Formulation

$$\max_{\alpha \in \mathbb{R}^N} 2 \sum_{i=1}^{N} \alpha_i y_i - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K(x_i, x_j)$$

such that

$$0 \leq y_i \alpha_i \leq \frac{1}{2\lambda N} \text{ for } i = 1, \ldots, N$$

Pictorial Depiction for α values

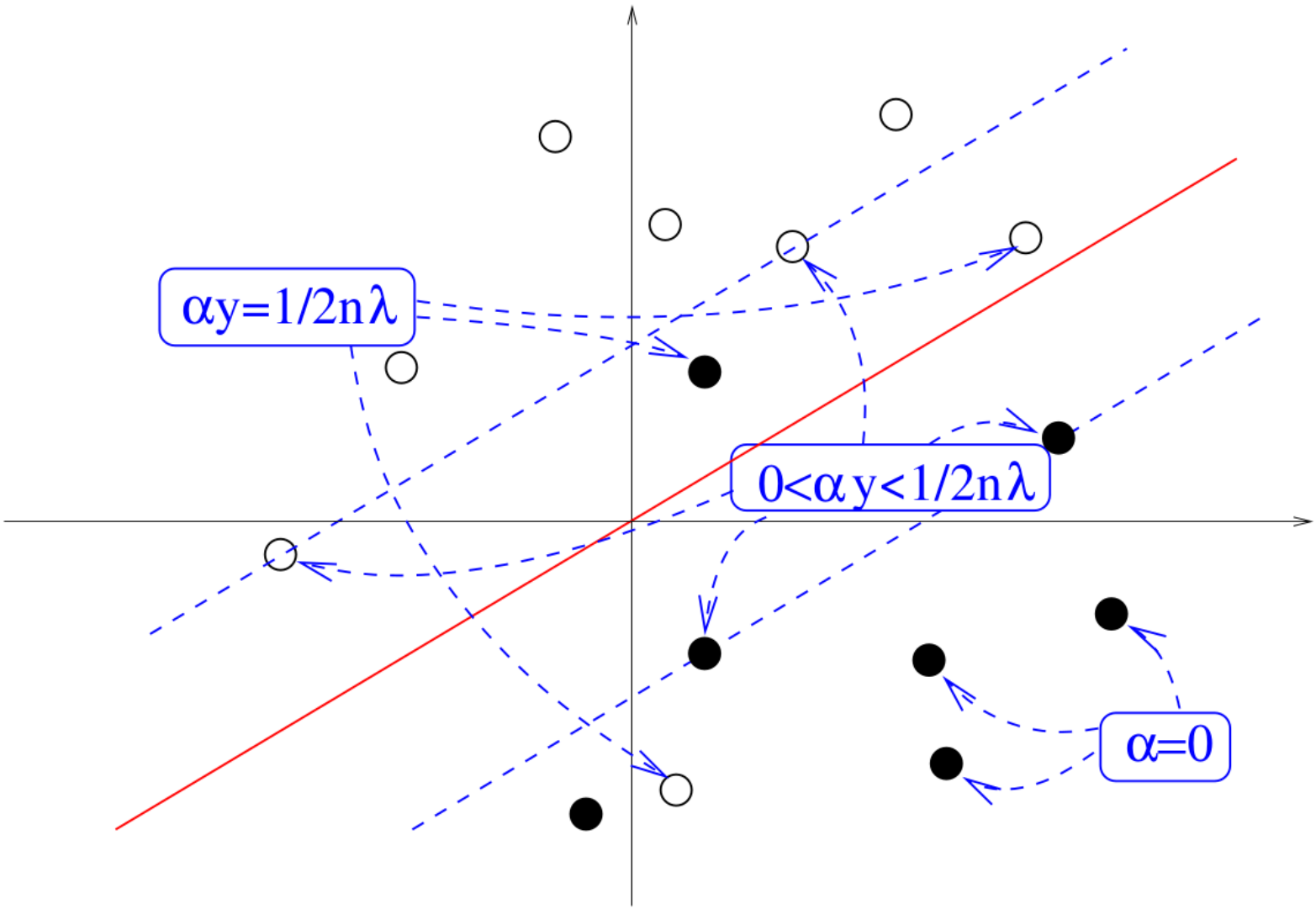


Figure: Training points with different values of α, (Picture : Julien Mairal)

Weighted Regression

- In ridge regression, we weight each error uniformly
- Suppose, we weigh the error at each training point differently, such that $\beta_i > 0$ is weight of error at point i, then
- The corresponding objective function is

$$\arg\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \beta_i (y_i - f(x_i))^2 + \lambda ||f||_{\mathcal{H}}$$

- How do we solve it?
- Using Representer Theorem, noticing that solution is of the form $\sum_{i=1}^{N} \alpha_i K(x_i, .)$, where is obtained by solving the following :

$$\arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \frac{1}{N} (K\boldsymbol{\alpha} - y)^T B (K\boldsymbol{\alpha} - y) + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

where B is a diagonal matrix with weight β_i at the $i-$th diagonal entry

Solving Kernel Logistic Regression

- By representer theorem, any solution to kernel logistic regression is given by

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

- Also, we have the following :
  - For the input instance, the prediction by the desired function can be written as follows :

$$(\hat{f}(x_1), \ldots, \hat{f}(x_N))^T = K\boldsymbol{\alpha}$$

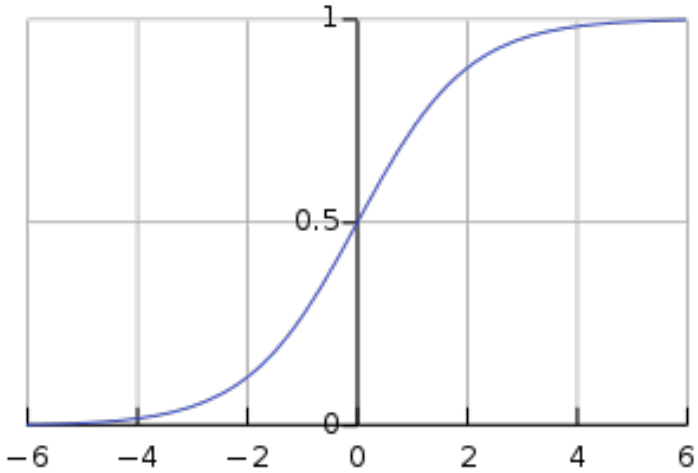
  - We also know that

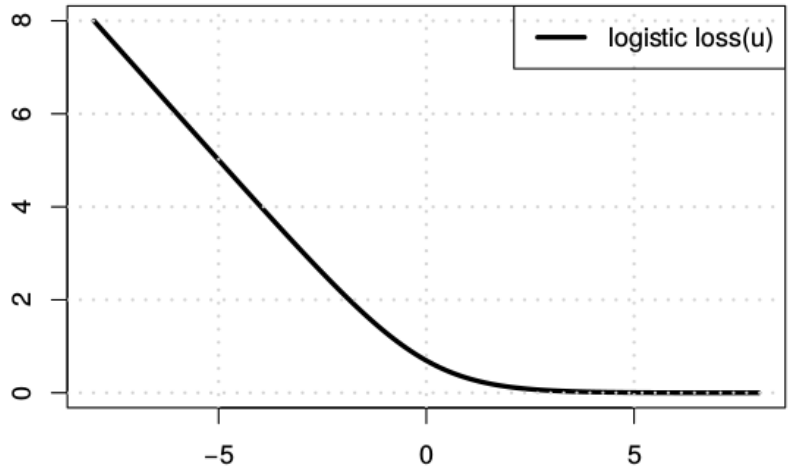
$$||f||_{\mathcal{H}}^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

- Therefore, we need to solve the following :

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-y_i [K\boldsymbol{\alpha}]_i)) + \frac{\lambda}{2} \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

Some facts related to Sigmoid and logistic loss





Sigmoid Function

- $\sigma(z) = \frac{1}{1+\exp(-z)}$
- $\sigma(-z) = 1 - \sigma(z)$
- $\sigma'(z) = \sigma(z)\sigma(-z) \geq 0$

Logistic loss

- $\ell_{logistic}(z) = \log(1 + \exp(-z))$
- $\ell'_{logistic}(z) = -\sigma(-z)$
- $\ell''_{logistic}(z) = \sigma(z)\sigma(-z) \geq 0$

Dimensionality reduction

- Motivation: High-dimensional data, where *interesting* pattern concerns a simpler subspace with much lower dimensionality
- Dimensionality reduction: Given D dimensional dataset $\{x_i \in \mathbb{R}^D\}_{i=1}^N$, find a low-dimensional representation $\{z_i \in \mathbb{R}^d\}_{i=1}^N$

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \quad \Rightarrow \quad Z = \begin{bmatrix} z_{11} & z_{12} & \dots & z_{1d} \\ z_{21} & z_{22} & \dots & z_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ z_{N1} & z_{N2} & \dots & z_{Nd} \end{bmatrix}$$

where $d \ll D$

Principal component analysis - Optimization

- Let us write PCA as an optimization problem

$$\max_{w_1 \in \mathbb{R}^D} \; w_1^T \Sigma_X w_1$$
$$\text{s.t.} \, w_1^T w_1 = 1$$

- Constraints set w_1 to unit norm to prevent an unbounded solution
- The above is problem of variance maximization over w_1
  - What is a reasonable **lower bound** on the objective ?
- Is it convex or non-convex optimization problem?
- This is a **non-convex optimization problem** (feasible set is non-convex + **maximizing** a convex objective)
- From KKT conditions, we can still use the Lagrangian approach to find necessary (but not sufficient) conditions for optimality

Eigen-decompositions of covariance and kernel matrix

- Consider the eigenvalue decompositions of the sample covariance matrix $N \times \Sigma_X = X^T X = U\tilde{\Lambda}_D U^T$ and the kernel matrix $K = XX^T = V\Lambda_n V^T$ where $\Sigma_X = \left(\frac{1}{N}\sum_{i=1}^{N} x_i {x_i}^T\right)$
- For an eigenvector-eigenvalue pair (v, λ) of the kernel matrix:

$$\begin{aligned} Kv = \lambda v \Rightarrow X^T Kv &= X^T \lambda v \\ \Rightarrow X^T X(X^T v) &= \lambda(X^T v) \\ \Rightarrow N \times \Sigma_X(X^T v) &= \lambda(X^T v) \end{aligned}$$

- Thus, $(X^T v, \lambda)$ is an eigenvector-eigenvalue pair of $N \times \Sigma_X$, in other words a principal component,
- The principal component is expressed as a linear combination $X^T v = \sum_{i=1}^{N} v_i x_i$ (note : $x_i \in \mathbb{R}^D$ and $v_i \in \mathbb{R}$) of data points (similar to Representer Theorem [1])

[1] More details - A Generalized Representer Theorem, COLT 2000, Schoelkopf etal.

Projections using kernels

- Since $\left\|X^T v\right\|^2 = v^T X X^T v = v^T (Kv) = v^T (\lambda v) = v^T v \lambda = \lambda$, the normalised eigenvector is given by

$$u = \frac{X^T v}{\|X^T v\|} = \lambda^{-1/2} X^T v$$

- Writing $X^T v$ in terms of the feature vectors we get:

$$u = \frac{X^T v}{\|X^T v\|} = \lambda^{-1/2} X^T v = \lambda^{-1/2} \sum_{i=1}^{N} x_i v_i$$

Projections using kernels

- The eigenvectors of the sample covariance matrix can be written in dual form as

$$u_j = \sum_{i=1}^{N} \alpha_i^j x_i, j = 1, \ldots, d$$

  where the vector of dual variables satisfies $\alpha^j = \lambda_j^{-1/2} v_j$, and v_j is the j'th eigenvector of the kernel matrix
- Thus we can compute the projection in the direction u_j using kernels

$$P_{u_j}(x) = u_j^T x = \langle \sum_{i=1}^{N} \alpha_i^j x_i, x \rangle$$

$$= \sum_{i=1}^{N} \alpha_i^j k(x_i, x)$$

Course summary

Broadly topics covered in this course

- Basics about Kernels
  - Reproducing Kernel Hilbert Space
  - Representer theorem
- Foundational learning theory
- Convex optimization overview
- Supervised learning algorithms with Kernels
  - Ridge regression
  - Logistic regression
  - Support vector machines
- Unsupervised learning algorithms with Kernels
  - Principal Component Analysis
  - Clustering
- Kernels for structured and multi-view data
- Bochner's theorem

Thank you for following the course!