CS6208 : Advanced Topics in Artificial Intelligence
Graph Machine Learning

Lecture 4 : Graph SVM

Semester 2 2022/23

Xavier Bresson

https://twitter.com/xbresson

Department of Computer Science

National University of Singapore (NUS)

Course lectures

- Introduction to Graph Machine Learning
- Part 1: GML without feature learning (before 2014)
  - Introduction to Graph Science
  - Graph Analysis Techniques without Feature Learning
    - Graph clustering
    - Graph SVM
    - Recommendation
    - Dimensionality reduction
- Part 2 : GML with shallow feature learning (2014-2016)
  - Shallow graph feature learning
- Part 3 : GML with deep feature learning, a.k.a. GNNs (after 2016)
  - Graph Convolutional Networks (spectral and spatial)
  - Weisfeiler-Lehman GNNs
  - Graph Transformer & Graph ViT/MLP-Mixer
  - Benchmarking GNNs
  - Molecular science and generative GNNs
  - GNNs for combinatorial optimization
  - GNNs for recommendation
  - GNNs for knowledge graphs
  - Integrating GNNs and LLMs

Outline

- Supervised classification
- Linear SVM
- Soft-margin SVM
- Kernel techniques
- Non-linear/kernel SVM
- Graph SVM
- Conclusion

Outline

- **Supervised classification**
- Linear SVM
- Soft-margin SVM
- Kernel techniques
- Non-linear/kernel SVM
- Graph SVM
- Conclusion

Learning techniques

- As of Feb 2023, there are five main classes of learning algorithms :
  - Supervised learning (SL) : Algorithms that use labeled data, i.e. data annotated by humans.
  - Unsupervised learning : Algorithms that learn the underlying data distribution without relying on label information, e.g. data generation.
  - Semi-supervised learning : Algorithms that use both labeled and unlabeled data.
  - Reinforcement learning (RL) : Algorithms that learn sequence of actions to maximize a future reward over time, e.g. winning games.
  - Self-supervised learning (SSL) : Algorithms that learn data representation by self-labeling, without requiring human annotations.

Support vector machine

- In this lecture, we will focus on two specific topics :
  - Supervised classification using Support Vector Machine (SVM).
  - Semi-supervised learning that leverage graph structure to improve learning from partially labeled data.
- SVM stands as a theoretically robust and widely successful technique deployed across various applications.
- It was the prevailing machine learning model prior to the advent of deep learning.
- Its decline in popularity can be attributed primarily to the absence of a feature learning mechanism. SVM relies on features engineered by humans, which were surpassed with features learned by neural network architectures.

Support vector machine

- SVM elegantly connects important topics in machine learning :
  - Geometric interpretation of classification tasks.
  - Ability to handle non-linear class boundaries using higher-dimensional feature maps.
  - Efficient use of the kernel trick to maintain the complexity of input data.
  - High-dimensional interpolation with the representer theorem.
  - Use of graph representations to capture data distribution regardless of labels.
  - Incorporation of graph regularization to propagate label information throughout the graph domain.
  - Primal and dual optimization methods for solving quadratic programming problems.

Outline

- Supervised classification
- Linear SVM
- Soft-margin SVM
- Kernel techniques
- Non-linear/kernel SVM
- Graph SVM
- Conclusion

SVM formulation

Vladimir Vapnik

- **Goal : Given a set V of labeled data with two classes, the goal is to construct a classification function f that assigns the class for new, previously unseen data point by maximizing the margin between the two classes[1].**

$$f : x \in \mathbb{R}^d \to \{-1, 1\}$$

$$\text{with } V = \{x_i, \ell_i\}_{i=1}^n, x_i \in \mathbb{R}^d \quad \text{(data features)}$$

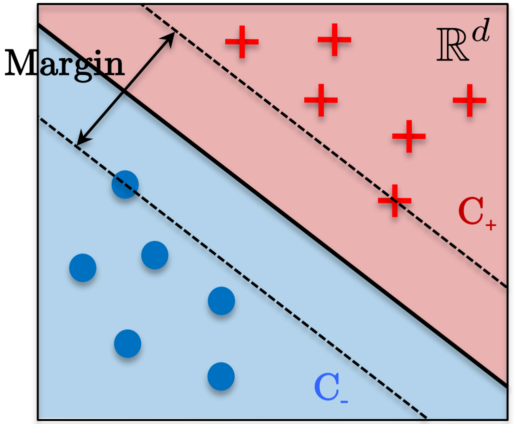
$$\ell_i \in \{-1, 1\} \text{ (data label)}$$

Positive label : +

$x_i, \ell_i = +1$

Classification function :

$f(x) = +1, \ x \in C_+$



Negative label :

$x_i, \ell_i = -1$

Classification function :

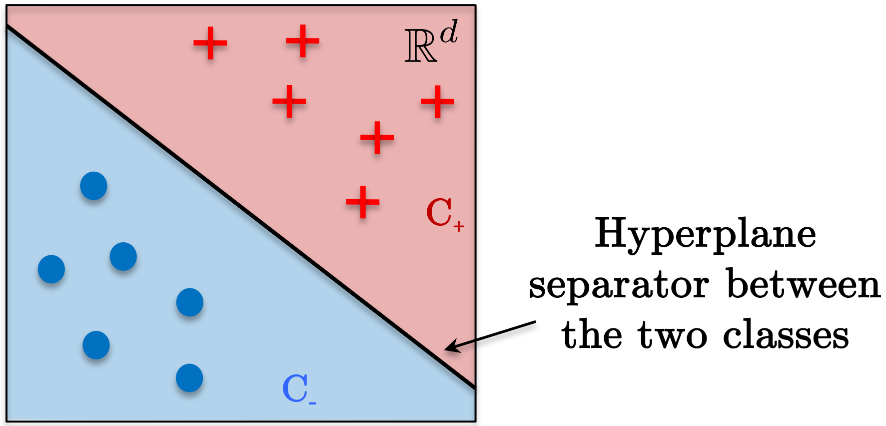
$f(x) = -1, \ x \in C_-$

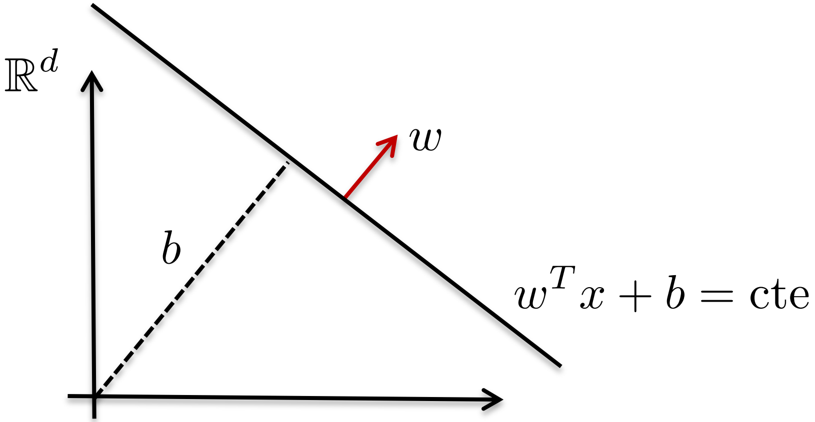
[1] Vapnik, Chervonenkis, On a perceptron class, 1964

Linear SVM

- Assumption[1] : Training and test datasets are linearly separable, i.e. data can be separated with a straight line in 2D, a plane in 3D and a hyper-plane in higher dimensions.
- A hyper-plane is parameterized with two variables (w, b), where w is the normal vector of the hyper-plane, i.e. determining its slope, and b is the offset or bias term :

$$\text{Hyper-plane equation : } \{x : w^T x + b = \text{cte}\},\ \ x, w \in \mathbb{R}^d,\ \ b \in \mathbb{R}$$



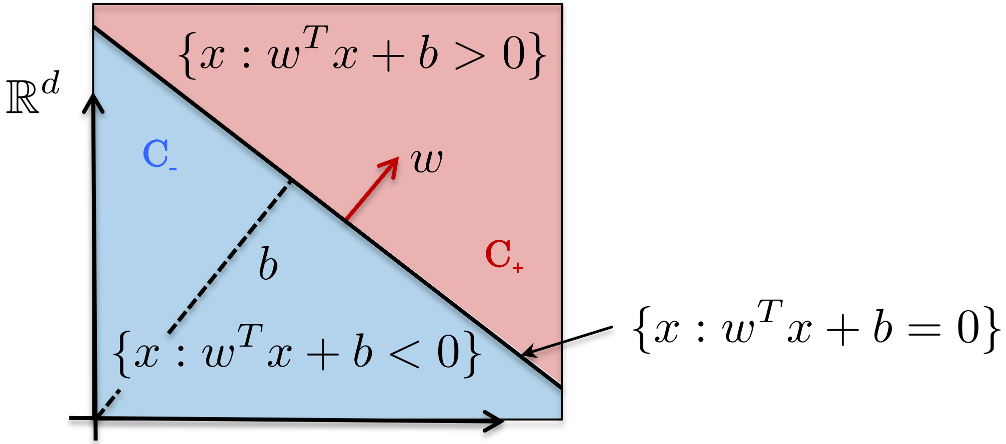


[1] Vapnik, Chervonenkis, On a perceptron class, 1964

SVM classifier

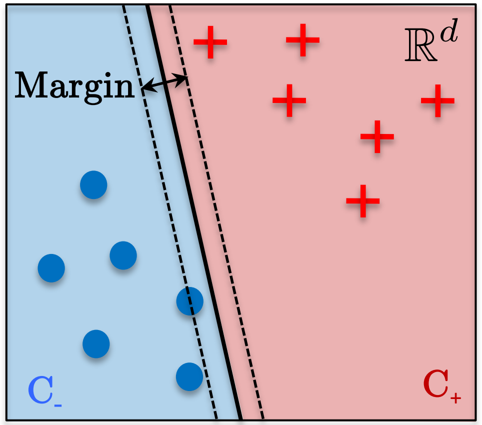
- **Classification function :**

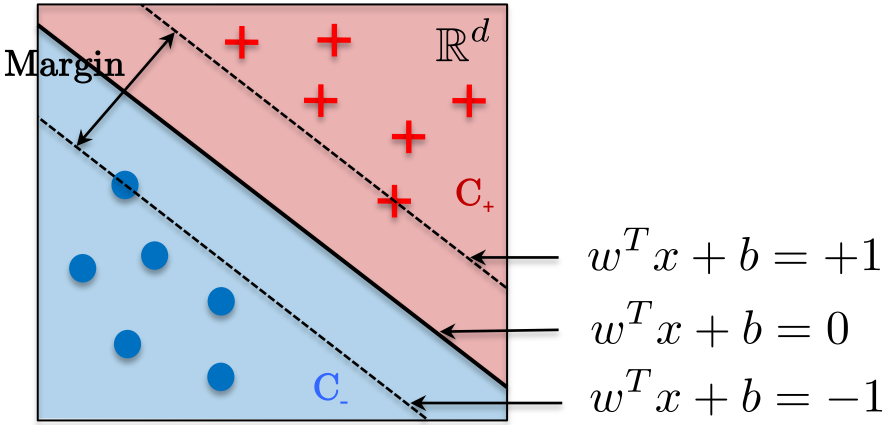
$$f_{w,b}(x) = \text{sign}(w^T x + b) = \begin{cases} +1 & \text{for } x \in C_+ \\ -1 & \text{for } x \in C_- \end{cases}$$



Maximizing class margin

- Hyper-plane $w^T x + b = 0$ is the class separator.
- Hyper-planes $w^T x + b = \pm 1$ are the class margins.
- Why do we want to maximize the margin?
  - Note that multiple hyper-plane solutions exist to separate the two classes.
  - Let us select the solution that generalizes the best, i.e. the solution with the largest margin between the classes.





Maximizing class margin

- **What are the parameters (w, b) that maximize the margin d between the training points?**

Margin is defined with the vector $d = x_+ - x_- \in \mathbb{R}^d$

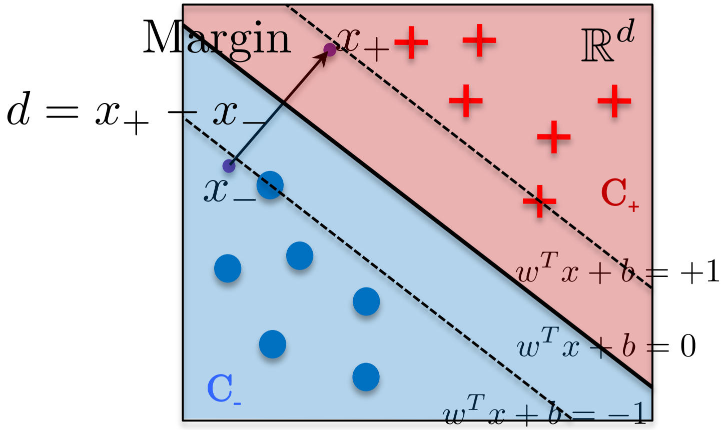
Given that $w^T x_+ + b = +1$ and $w^T x_- + b = -1$ and

substracting these two lines : $w^T(x_+ - x_-) = 2 \;\Rightarrow\; w^T d = 2$

Then taking the norm : $\|w\|_2.\|d\|_2 = 2 \;\Rightarrow\; \|d\|_2 = \dfrac{2}{\|w\|_2}$

Finally, $\max\limits_{d} \; \|d\|_2 = \dfrac{2}{\|w\|_2} \;\Leftrightarrow\; \min\limits_{w} \; \|w\|_2^2$ s.t. $\begin{cases} w^T x_i + b \geq +1 & \text{if } x_i \in C_+ \\ w^T x_i + b \geq -1 & \text{if } x_i \in C_- \end{cases}$

Maximizing the class margin is equivalent to minimize the norm of w while satisfying the label constraints.



Primal optimization problem

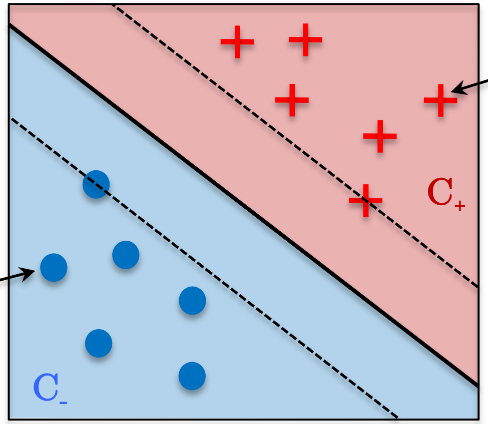
- **Optimal value w^* is the solution of a constrained quadratic programming (QP) problem :**

$$\min_w \ \|w\|_2^2 \ \text{ s.t. } \ \ell_i . s_i \geq 1, \ \forall i \in V$$

$$\text{with } \ s_i = w^T x_i + b = \begin{cases} \geq +1 & \text{for } x \in C_+ \\ \leq -1 & \text{for } x \in C_- \end{cases} \quad \text{and} \quad \ell_i = \begin{cases} +1 & \text{if } x \in C_+ \\ -1 & \text{if } x \in C_- \end{cases}$$

which can be compactly expressed as $\ell_i . s_i \geq 1, \ \forall i \in V$

$s_i = w^T x_i + b \geq +1$
$\ell_i = +1$



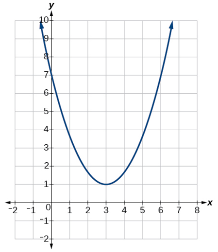
$s_i = w^T x_i + b \leq -1$
$\ell_i = -1$

Primal optimization problem

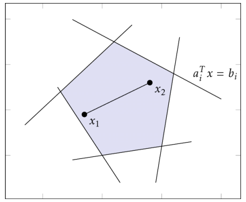
- There exists a unique solution to the QP[1,2,3] optimization problem, if the assumption of linearly separable data points is satisfied.
- Variable w is called the primal variable.

$$w^\star = \arg\min_w \; \|w\|_2^2 \quad \text{s.t.} \quad \ell_i . s_i \geq 1, \; \forall i \in V \qquad \Rightarrow \qquad f_{\text{SVM}}(x) = \text{sign}((w^\star)^T x + b^\star)$$

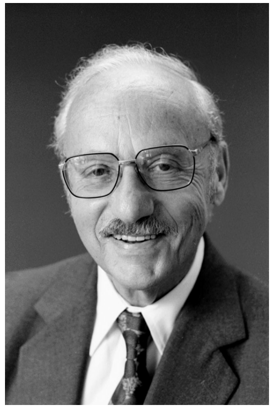
Quadratic function



Convex set (polytope)



SVM classifier



George Dantzig
1914-2005

[1] Dantzig, Orden, Wolfe, The generalized simplex method for minimizing a linear form under linear inequality restraints, 1955
[2] Wolfe, The Simplex Method for Quadratic Programming, 1959
[3] Boyd, Vandenberghe, Convex Optimization, 2004

Support vectors

- **Support vectors are the data points exactly localized on the margin hyper-planes :**

$$\ell_i . s_i = 1, \;\; \forall x_i^{\text{sv}} \text{ (support vectors)}$$

$$\ell_i . ((w^\star)^T x_i^{\text{sv}} + b^\star) = 1$$

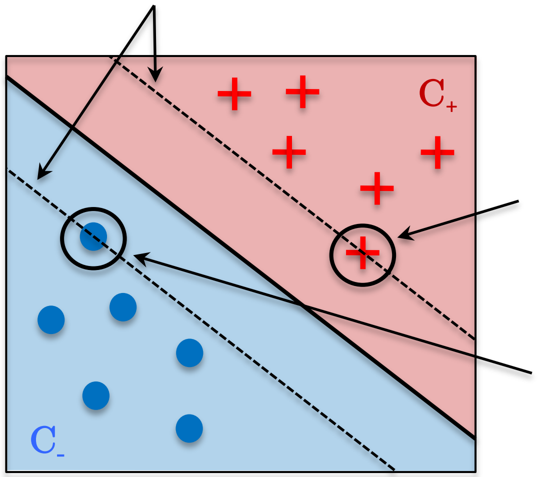
which gives

$$b^\star = \ell_i - (w^\star)^T x_i^{\text{sv}}$$

and on expectation

$$b^\star = \frac{1}{|x_i^{\text{sv}}|} \sum_{x_i^{\text{sv}}} \ell_i - (w^\star)^T x_i^{\text{sv}}$$

Margin planes



Support vectors

Dual variable

- We can represent the weight vector w as a linear combination $\propto$ of the training data points x_i.
- The coefficient vector $\propto$ is referred to as the dual variable of w.
- The dual problem naturally introduces the linear kernel matrix $K(x, y) = x^T y$:

$$
\begin{aligned}
\text{Given } w &= \sum_i \alpha_i \ell_i x_i \in \mathbb{R}^d, \alpha_i \in \mathbb{R} \\
\text{we have } w^T x &= \sum_i \alpha_i \ell_i x_i^T x \in \mathbb{R} \\
&= \sum_i \alpha_i \ell_i K(x_i, x) \text{ with } K(x_i, x) = x_i^T x \\
&= \alpha^T L K(x), \ \alpha, K(x) \in \mathbb{R}^n, L \in \mathbb{R}^{n \times n}
\end{aligned}
$$

$$
\begin{aligned}
\text{Classification function : } \quad f_{\text{SVM}}(x) &= \text{sign}(w^T x + b) \in \pm 1 \qquad \text{(with primal variable)} \\
&= \text{sign}(\alpha^T L K(x) + b) \in \pm 1 \quad \text{(with dual variable)}
\end{aligned}
$$

Dual optimization problem

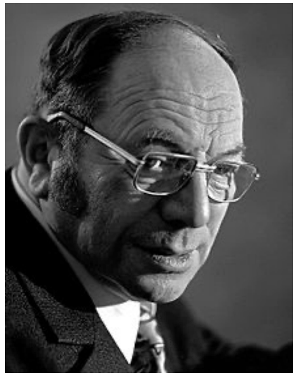
- **The primal optimization problem can be solved with the dual problem[1,2,3] :**

$$\min_{w} \ \|w\|_2^2 \ \text{ s.t. } \ \ell_i.s_i \geq 1, \ \forall i \in V \quad \text{(primal QP problem)}$$

is equivalent to

$$\min_{\alpha \geq 0} \ \frac{1}{2}\alpha^T Q \alpha - \alpha^T 1_n \ \text{ s.t. } \ \alpha^T \ell = 0 \ \ \text{(dual QP problem)}$$

$$\begin{aligned} \text{with } Q &= LKL \in \mathbb{R}^{n\times n} \\ L &= \text{diag}(\ell) \in \mathbb{R}^{n\times n} \\ \ell &= (\ell_1, ..., \ell_n) \in \mathbb{R}^n \\ K &\in \mathbb{R}^{n\times n}, K_{ij} = x_i^T x_j \in \mathbb{R} \ \ \text{(linear kernel)} \end{aligned}$$



Leonid Kantorovich
1912-1986

[1] Kantorovich, The Mathematical Method of Production Planning and Organization, 1939
[2] Dantzig, Orden, Wolfe, The generalized simplex method for minimizing a linear form under linear inequality restraints, 1955
[3] Boyd, Vandenberghe, Convex Optimization, 2004

Optimization algorithm

- **Solution α^* can be computed with a simple primal-dual[1,2] iterative scheme :**

Narendra Karmarkar

Initialization : $\alpha^{k=0} = \beta^{k=0} = 0_n \in \mathbb{R}^n$

Time steps satisfy $\tau_\alpha \tau_\beta \leq \dfrac{1}{\|Q\|\|L\|}$ s.a. $\tau_\alpha = \dfrac{1}{\|Q\|}, \tau_\beta = \dfrac{1}{\|L\|}$

Iterate :

$$\alpha^{k+1} = \mathrm{P}_{\cdot \geq 0}\big((\tau_\alpha Q + \mathrm{I}_n)^{-1}(\alpha^k + \tau_\alpha Q - \tau_\alpha L \beta^k)\big) \in \mathbb{R}^n$$

$$\beta^{k+1} = \beta^k + \tau_\beta L \alpha^{k+1} \in \mathbb{R}^n$$

At convergence, we have : $\alpha^\star$

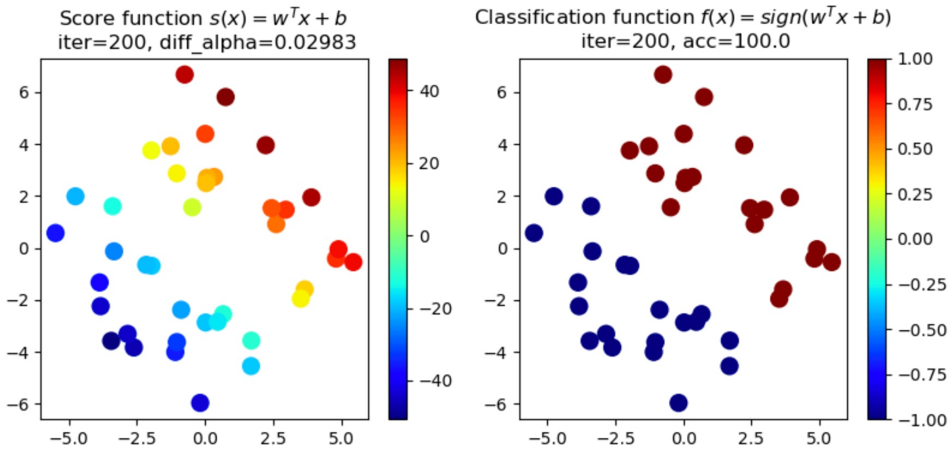
Classification function : $f_{\mathrm{SVM}}(x) = \mathrm{sign}(\alpha^{\star T} L K(x) + b^\star) \in \pm 1$

[1] Karmarkar, A new polynomial-time algorithm for linear programming, 1984
[2] Boyd, Vandenberghe, Convex Optimization, 2004

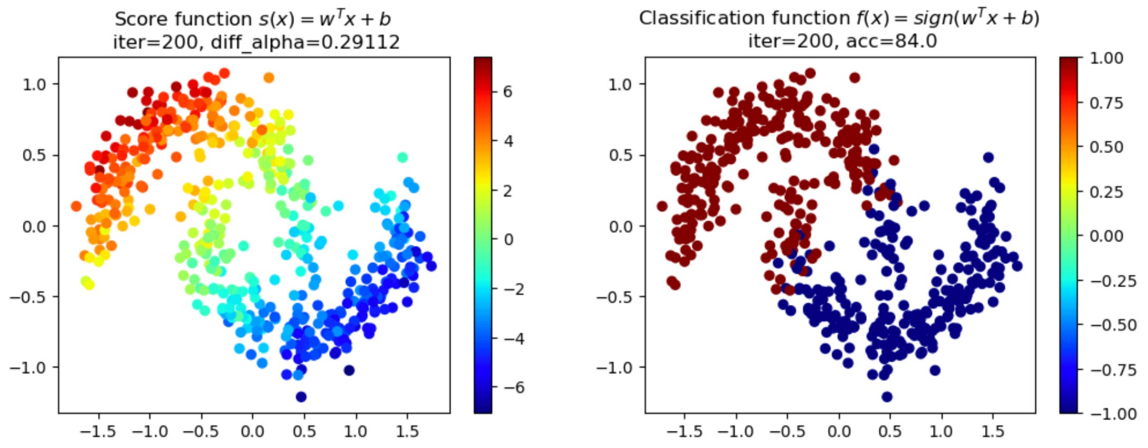
Lab 1 : Linear SVM

- Run code01.ipynb and analyze linear SVM result on
  - Linearly separable data points
  - Non-linear data points

```
•[5]: # Run Linear SVM

# Compute linear kernel, L, Q
Ker = Xtrain.dot(Xtrain.T)
l = l_train
L = np.diag(l)
Q = L.dot(Ker.dot(L))

# Time steps
tau_alpha = 1./ np.linalg.norm(Q,2)
tau_beta = 1./ np.linalg.norm(L,2)

# For conjuguate gradient
Acg = tau_alpha* Q + np.eye(n)

# Pre-compute J.K(Xtest) for test data
LKXtest = L.dot(Xtrain.dot(Xtest.T))

# Initialization
alpha = np.zeros([n])
beta = 0.0
alpha_old = alpha

# Loop
k = 0
diff_alpha = 1e6
num_iter = 101
while (diff_alpha>1e-3) & (k<num_iter):

    # Update iteration
    k += 1

    # Update alpha
    # Approximate solution with conjuguate gradient
    b0 = alpha + tau_alpha - tau_alpha* l* beta
    alpha, _ = scipy.sparse.linalg.cg(Acg, b0, x0=alpha, tol=1e-3, maxiter=50)
    alpha[alpha<0.0] = 0 # Projection on [0,+infty]

    # Update beta
    beta = beta + tau_beta* l.T.dot(alpha)

    # Stopping condition
    diff_alpha = np.linalg.norm(alpha-alpha_old)
    alpha_old = alpha
```



Linearly separable data points



Non-linear data points

Outline

- Supervised classification
- Linear SVM
- Soft-margin SVM
- Kernel techniques
- Non-linear/kernel SVM
- Graph SVM
- Conclusion

Noise

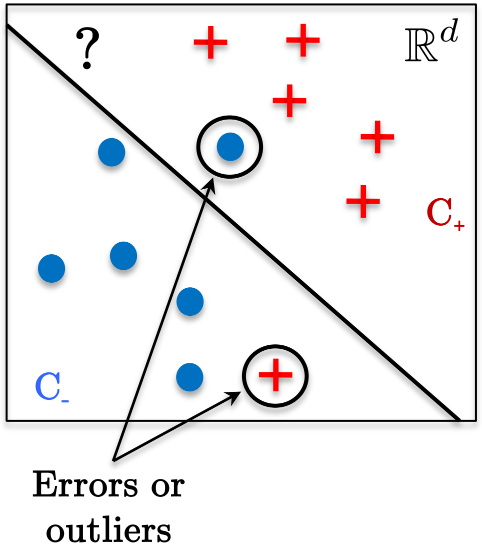
- Real-world data often contains noise and outliers, which do not satisfy the assumption of linearly separable data points.
- When dealing with non-linearly separable data, there is no mathematical solution for standard or hard-margin SVM because there does not exist a linear separator that can split the two classes perfectly, i.e. without errors.
- A new technique is necessary, refered as soft-margin SVM[1].

Positive label : +

$x_i, \ell_i = +1$

Classification function :

$f(x) = +1, \ x \in C_+$



Negative label : ●

$x_i, \ell_i = -1$

Classification function :

$f(x) = -1, \ x \in C_-$

[1] Cortes, Vapnik, Support-vector networks, 1995

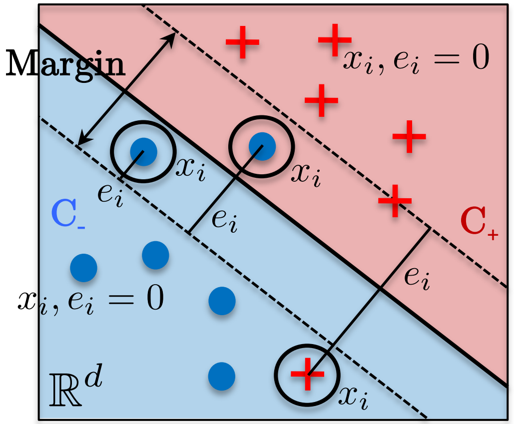
Soft-margin SVM

- Slack variables e_i quantifies the error for each data x_i to be an outlier.
- These errors e_i will be minimized while simultaneously maximizing the margin :

$$\min_{w} \ \|w\|_2^2 \ \text{ s.t. } \begin{cases} w^T x_i + b \geq +1 & \text{for } x_i \in C_+ \\ w^T x_i + b \leq -1 & \text{for } x_i \in C_- \end{cases} \quad \text{(Standard SVM)}$$

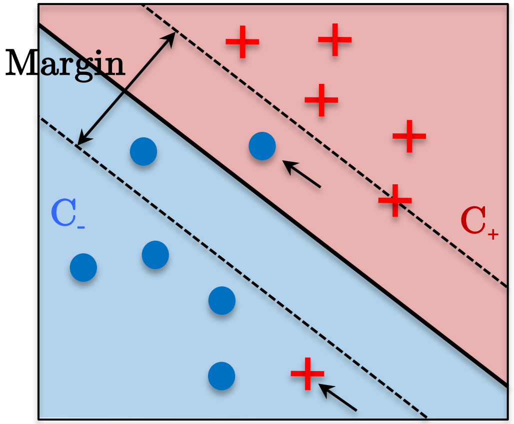
$$\Downarrow$$

$$\min_{w,e} \ \underbrace{\|w\|_2^2 + \lambda \sum_{i=1}^n e_i}_{\text{Trade-off between large margin and small errors}} \ \text{ s.t. } \begin{cases} w^T x_i + b \geq +1 - e_i & \text{for } x_i \in C_+ \\ w^T x_i + b \leq -1 + e_i & \text{for } x_i \in C_- \\ e_i \geq 0 & \text{for } x_i \in V \end{cases} \quad \text{(Soft-margin SVM)}$$

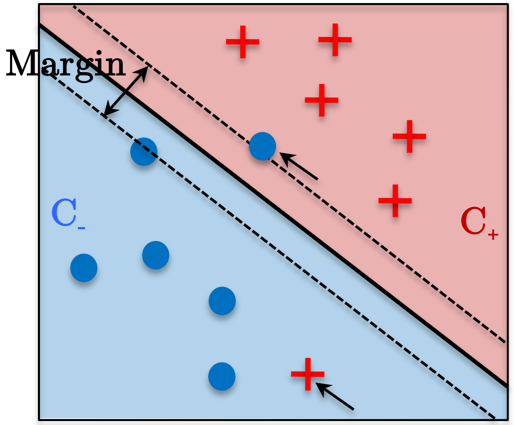


Regularization

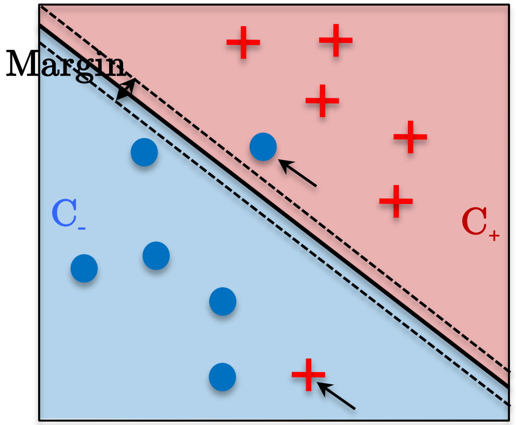
- What is the effect of varying λ, the regularization parameter?
  - For small λ values, more misclassification errors are allowed, the margin is larger.
  - For large λ values, misclassification errors are penalized, leading to either no errors or very few, resulting in a smaller margin.



Small λ value



Intermediate λ value



Large λ value

Hinge loss

- The soft-margin SVM technique penalizes :
  - Misclassifications of training data points.
  - Correct classifications of training points that fall inside the margin area.
- The constrained optimization problem can be reformulated as an unconstrained problem :

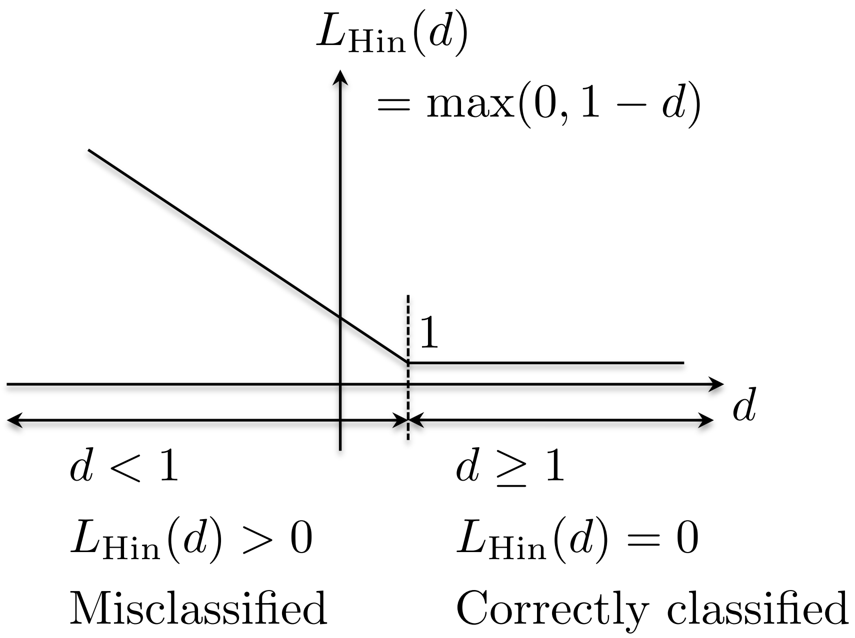
$$\min_{w,e} \|w\|_2^2 + \lambda \sum_{i=1}^{n} e_i \quad \text{s.t.} \quad \begin{cases} w^T x_i + b \geq +1 - e_i & \text{for } x_i \in C_+ \\ w^T x_i + b \leq -1 + e_i & \text{for } x_i \in C_- \\ e_i \geq 0 & \text{for } x_i \in V \end{cases}$$

$$\Updownarrow$$

$$\min_{w,e} \|w\|_2^2 + \lambda \sum_{i=1}^{n} \max\left(0, 1 - \ell_i s_i\right),$$

where $s_i = w^T x_i + b$ (score function)

$$L_{\text{Hin}}(d_i) = \max\left(0, 1 - d_i\right), \; d_i = \ell_i s_i \quad \text{(Hinge loss)}$$



Loss functions

- **There exist multiple loss functions[1] :**

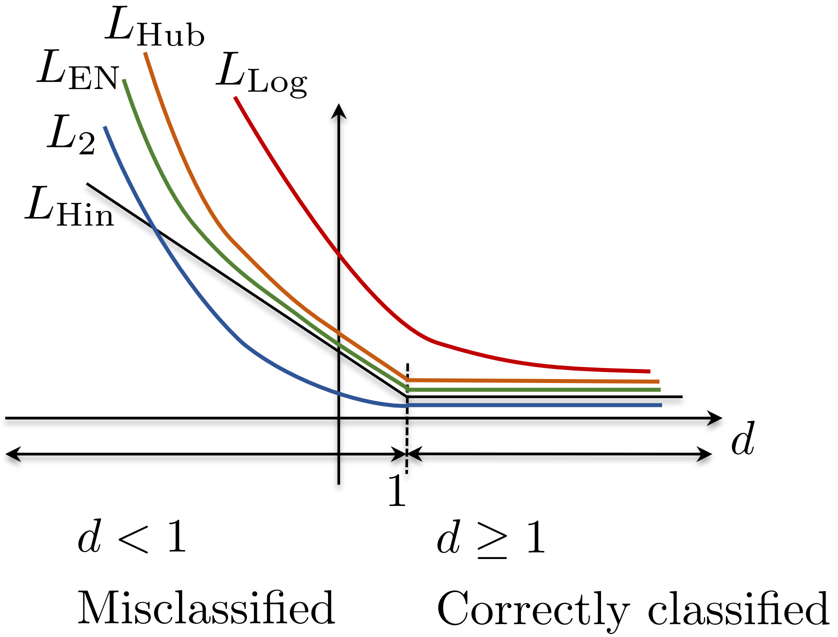
$$L_2(d_i) = \begin{cases} (1-d_i)^2 & \text{if } d_i < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{(L2 loss)}$$

$$L_{\text{EN}}(d_i) = \begin{cases} (1-d_i)^2 + \beta|1-d_i| & \text{if } d_i < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Elastic net loss)}$$

$$L_{\text{Hub}}(d_i) = \begin{cases} \frac{1}{2} - d_i & \text{if } d_i \leq 0 \\ \frac{1}{2}(1-d_i)^2 & \text{if } 0 < d_i < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Huber loss)}$$

$$L_{\text{Log}}(d_i) = \exp(1-d_i) \qquad \text{(Logistic loss)}$$

$$L_{\text{Hin}}(d_i) = \max(0, 1-d_i) \qquad \text{(Hinge loss)}$$



[1] Rosasco, De Vito, Caponnetto, Are loss functions all the same? 2004

Dual optimization problem

- **As previously, the primal optimization problem can be solved with the dual problem :**

$$\min_{w,e} \ \|w\|_2^2 + \lambda \sum_{i=1}^{n} e_i \ \text{ s.t. } \ \ell_i.s_i \geq 1 - e_i, \ e_i \geq 0 \ \forall i \in V \quad \text{(primal QP problem)}$$

is equivalent to

$$\min_{0 \leq \alpha \leq \lambda} \ \frac{1}{2} \alpha^T Q \alpha - \alpha^T 1_n \ \text{ s.t. } \alpha^T \ell = 0 \quad \text{(dual QP problem)}$$

Modification

$$\begin{aligned} \text{with } Q &= LKL \in \mathbb{R}^{n \times n} \\ L &= \text{diag}(\ell) \in \mathbb{R}^{n \times n} \\ \ell &= (\ell_1, ..., \ell_n) \in \mathbb{R}^n \\ K &\in \mathbb{R}^{n \times n}, K_{ij} = x_i^T x_j \in \mathbb{R} \quad \text{(linear kernel)} \end{aligned}$$

Optimization algorithm

- **Solution α* can be computed with the following iterative scheme :**

$$\text{Initialization}: \ \alpha^{k=0} = \beta^{k=0} = 0_n \in \mathbb{R}^n$$

$$\text{Time steps satisfy } \tau_\alpha \tau_\beta \leq \frac{1}{\|Q\|\|L\|} \ \text{ s.a. } \ \tau_\alpha = \frac{1}{\|Q\|}, \tau_\beta = \frac{1}{\|L\|}$$

Iterate :

$$\alpha^{k+1} = \mathrm{P}_{0 \leq \cdot \leq \lambda}\big((\tau_\alpha Q + \mathrm{I}_n)^{-1}(\alpha^k + \tau_\alpha Q - \tau_\alpha L \beta^k)\big)$$

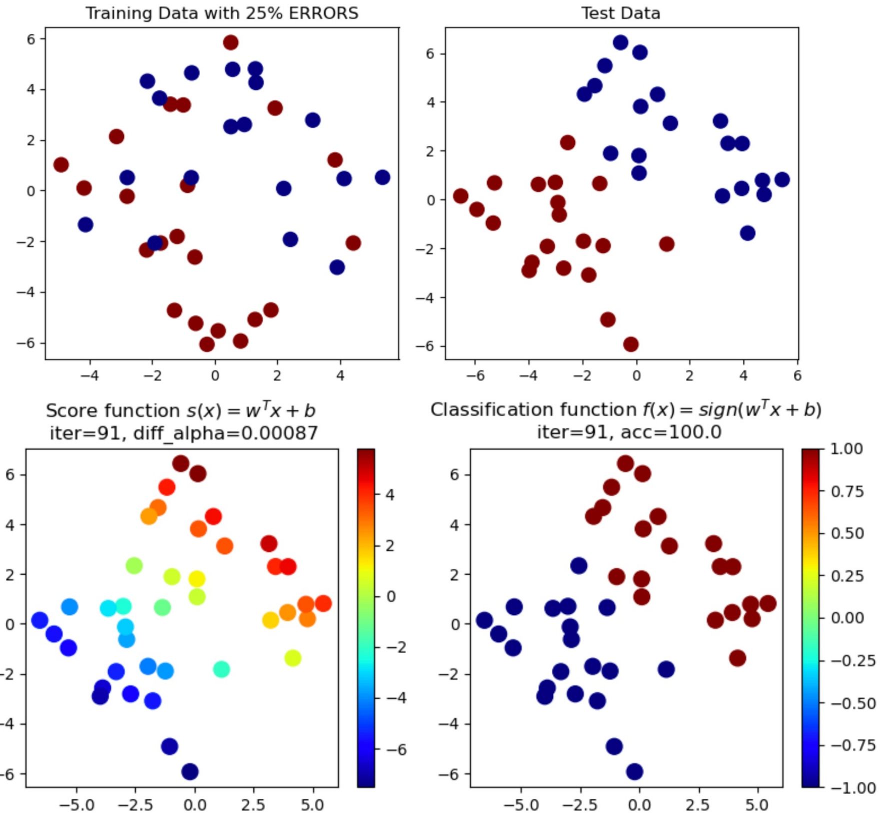
$$\beta^{k+1} = \beta^k + \tau_\beta L \alpha^{k+1}$$

$$\text{At convergence, we have}: \ \alpha^\star$$

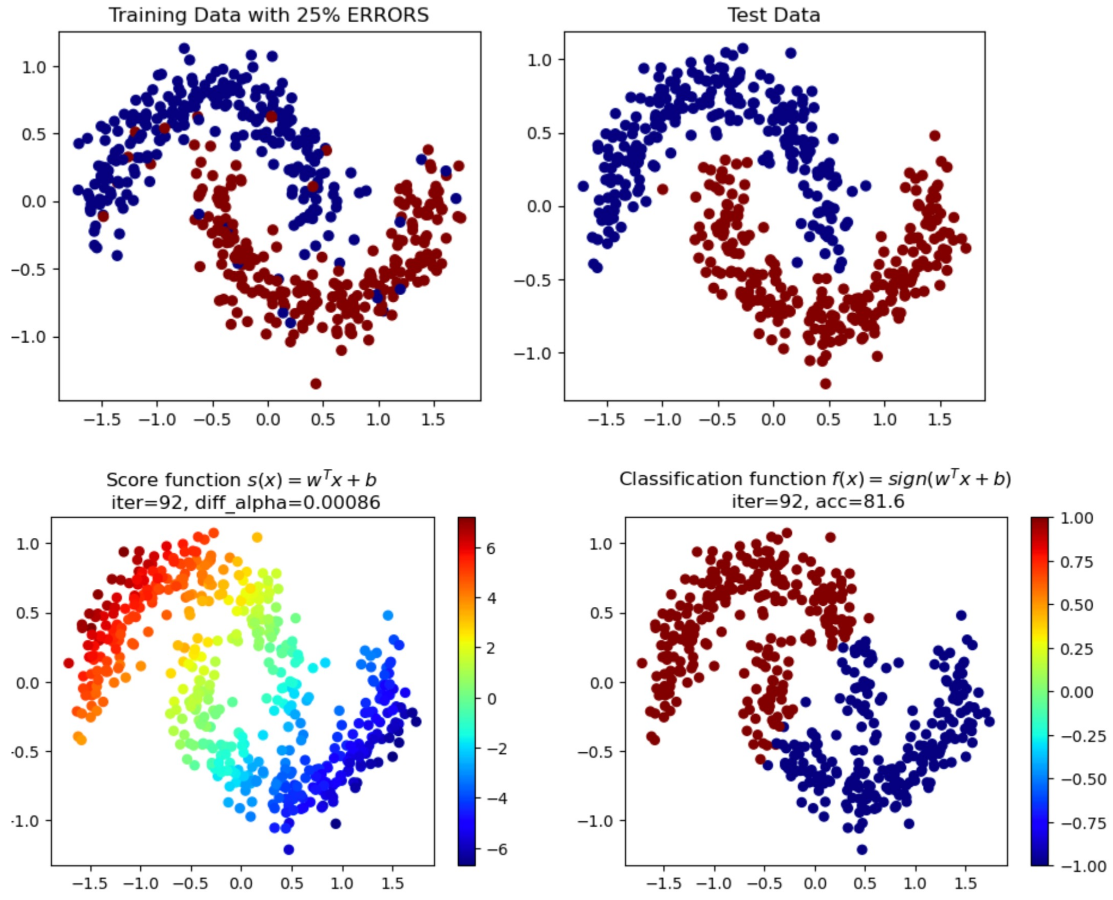
$$\text{Classification function}: \ \ f_{\text{SVM}}(x) = \text{sign}({\alpha^\star}^T L K(x) + b^\star) \in \pm 1$$

Lab 2 : Soft-margin SVM

- Run code02.ipynb and analyze SVM result on
  - Noisy linearly separable data points
  - Noisy non-linear data points



Noisy linearly separable data points



Noisy non-linear data points

Outline

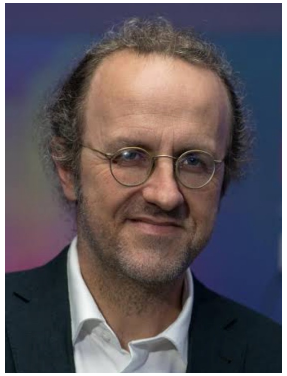
- Supervised classification
- Linear SVM
- Soft-margin SVM
- Kernel techniques
- Non-linear/kernel SVM
- Graph SVM
- Conclusion

High-dimensional interpolation

- How can we perform function interpolation in high-dimensional spaces?
- Reproducing Kernel Hilbert Space[1] (RKHS) : A space associated to bounded, symmetric, positive semidefinite (PSD) operator called a kernel $K(x, x) : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}_+$ that can reproduce any smooth function $h(x) : \mathbb{R}^d \longrightarrow \mathbb{R}$.
- Representer theorem[1,2] : Any continuous smooth function h in a RKHS can be represented as a linear combination of the kernel function K evaluated at the training data points x_i :

$$h(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b, \;\; x, x_i \in \mathbb{R}^d, b \in \mathbb{R} \;\; d \gg 1$$

David Hilbert
1862-1943

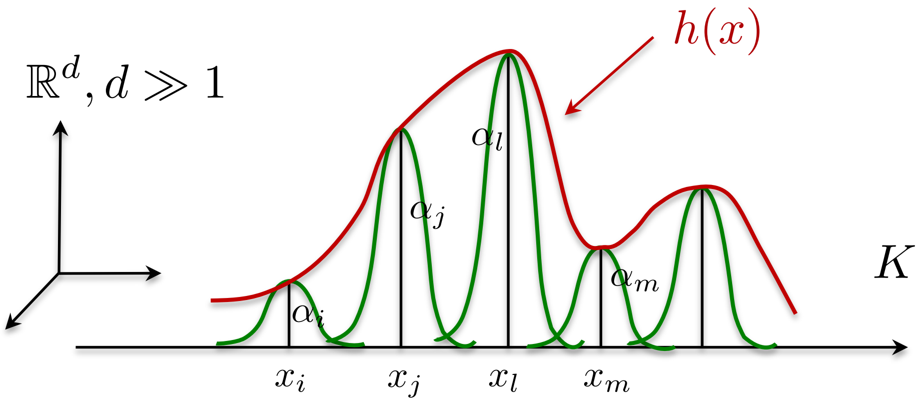


Bernhard
Schölkopf

[1] Beurling, On two problems concerning linear transformations in Hilbert space, 1948
[2] Scholkopf, Herbrich, Smola, A generalized representer theorem, 2001

Representer Theorem

- **Illustration of the Representer theorem to interpolate functions in high-dimensional spaces :**



$$K(x, x_i) = \exp(-\|x - x_i\|_2^2/\sigma^2)$$

$$h(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b \in \mathbb{R}$$

with the most common kernels are defined as

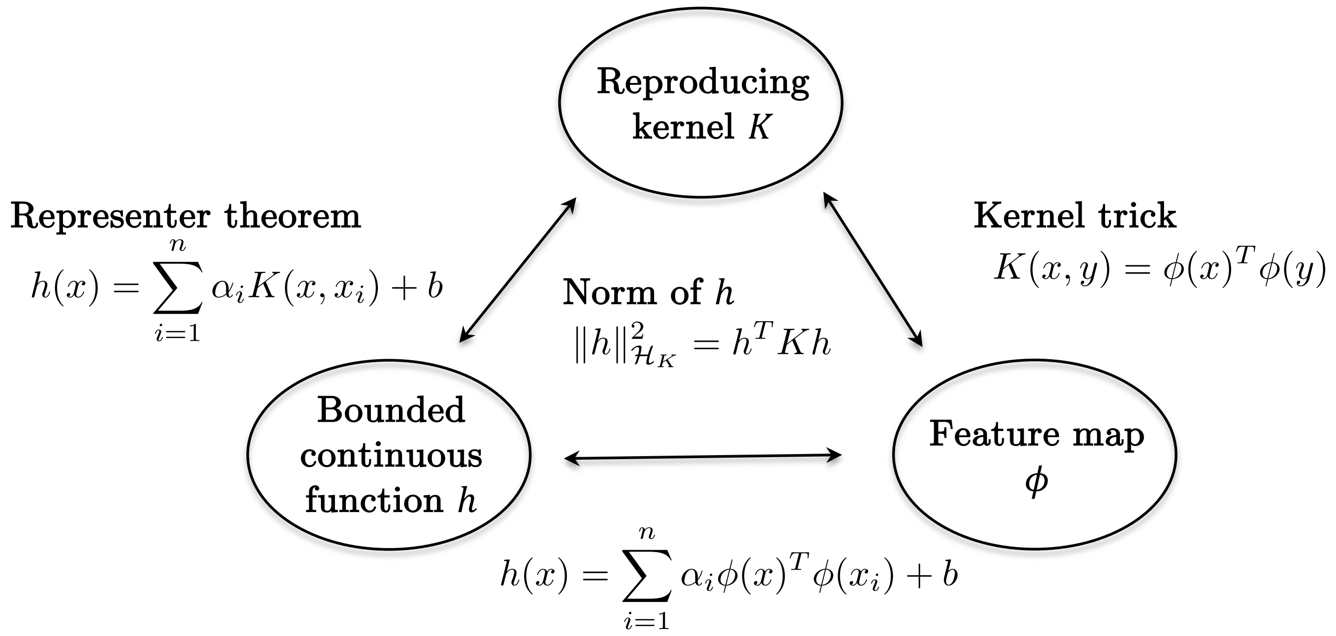
$$K(x, y) = x^T y \qquad \text{(linear kernel)}$$

$$K(x, y) = \exp(-\|x - y\|_2^2/\sigma^2) \qquad \text{(Gaussian kernel)}$$

$$K(x, y) = (ax^T y + b)^c \qquad \text{(polynomial kernel)}$$

Feature map, kernel trick and interpolation

- Any feature map ϕ defines a reproducing kernel K, and inversely.
- Any kernel K can be used to design a smooth high-dim function h.

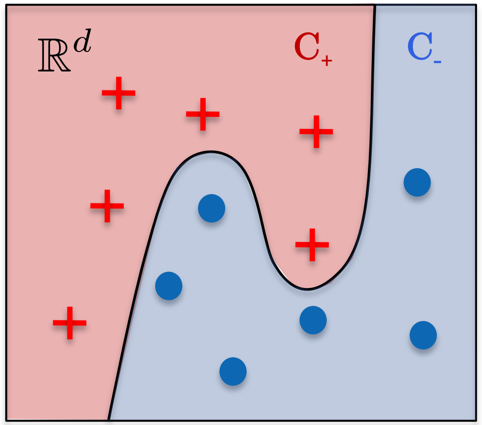


Outline

- Supervised classification
- Linear SVM
- Soft-margin SVM
- Kernel techniques
- Non-linear/kernel SVM
- Graph SVM
- Conclusion

Feature engineering for non-linear data

- Linear models, s.a. original and soft-margin SVM, assume linearly separable data points.
- But in many real-world scenarios, datasets are not linearly separable, i.e. a hyper-plane cannot distinguish between distinct classes.
- How to address this challenge and classify complex/non-linear datasets with linear separators?
- Feature engineering approach[1] : Project the data into a higher-dimensional space using a feature map ϕ where the data becomes linearly separable.

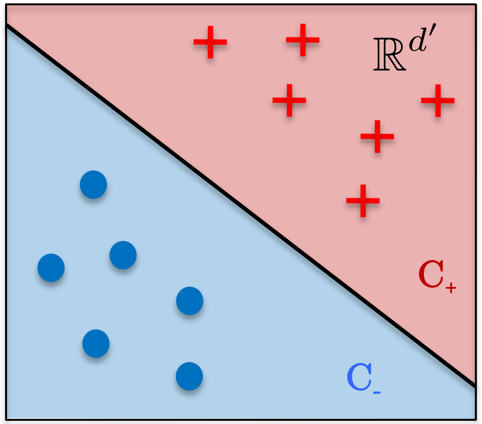


Non-linear dataset and separator

$$\xrightarrow{\text{Feature map } \phi(\cdot)}$$

$$x \in \mathbb{R}^d \ \rightarrow \phi(x) \in \mathbb{R}^{d'}$$

$$d' > d \text{ (possibly } d' \gg d)$$



Linear dataset and separator

[1] Aizerman et-al, Theoretical foundations of the potential function method in pattern recognition learning, 1964

Kernel trick

- Non-linear mapping ϕ enables the separation of non-linear data points.
- However, this approach entails operating in a larger feature space compared to the original one, resulting in increased complexity of $\mathrm{O}(d')$ where $\mathrm{d}' \gg \mathrm{d}$.
- To address this issue, the kernel trick was devised[1,2], offering a solution without the explicit use of the mapping ϕ.
  - With this approach, computing the kernel operator/matrix is defined as $\mathrm{K} = \phi^T\phi$, rather than ϕ individually, making the exact expression of ϕ irrelevant.
  - Some standard kernel operators include :

Time consuming

$$K(x_i, x_j) = x_i^T x_j \quad \text{(linear kernel for linear k-means)}$$

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) = \exp(-\|x_i - x_j\|_2^2/\sigma^2) \quad \text{(Gaussian kernel)}$$

$$K(x_i, x_j) = (a x_i^T x_j + b)^c \quad \text{(Polynomial kernel)}$$

Efficient kernel computation

[1] Aizerman et-al, Theoretical foundations of the potential function method in pattern recognition learning, 1964
[2] Guyon, Boser, Vapnik, Automatic capacity tuning of very large VC-dimension classifiers, 1993

Non-linear/kernel SVM

- **Primal optimization problem[1] w.r.t. w :**

$$\min_{w,e} \ \|w\|_2^2 + \lambda \sum_{i=1}^n e_i \ \text{ s.t. } \begin{cases} w^T\phi(x_i) + b \geq +1 - e_i & \text{for } x_i \in C_+ \\ w^T\phi(x_i) + b \leq -1 + e_i & \text{for } x_i \in C_- \\ e_i \geq 0 & \text{for } x_i \in V \end{cases} \quad \text{(Kernel SVM)}$$

- **Dual optimization problem w.r.t. α :**

$$\min_{0\leq\alpha\leq\lambda} \ \frac{1}{2}\alpha^T Q\alpha - \alpha^T 1_n \ \text{ s.t. } \alpha^T \ell = 0$$

$$\begin{aligned} \text{with } Q &= LKL \in \mathbb{R}^{n\times n} \\ L &= \text{diag}(\ell) \in \mathbb{R}^{n\times n} \\ \ell &= (\ell_1, ..., \ell_n) \in \mathbb{R}^n \\ K &\in \mathbb{R}^{n\times n}, K_{ij} = \phi(x_i)^T\phi(x_j) \in \mathbb{R} \quad \text{(generalized kernel)} \end{aligned}$$

Function ϕ is never used explicitly.

[1] Boser, Guyon, Vapnik, A training algorithm for optimal margin classifiers, 1992

Non-linear/kernel SVM

- **Decision function $f(x)$:**

$$
\begin{aligned}
\text{Given } w &= \sum_i \alpha_i \ell_i \phi(x_i) \in \mathbb{R}^d \\
\text{we have } w^T x &= \sum_i \alpha_i \ell_i \phi(x_i)^T \phi(x) \in \mathbb{R} \\
&= \sum_i \alpha_i \ell_i K(x_i, x) \text{ with } K(x_i, x) = \phi(x_i)^T \phi(x) \\
&= \alpha^T L K(x),\ \ \alpha, K(x) \in \mathbb{R}^n, L \in \mathbb{R}^{n\times n}
\end{aligned}
$$

$$
\begin{aligned}
\text{Classification function : } \quad f_{\text{SVM}}(x) &= \text{sign}(w^T \phi(x) + b) \quad \text{(with primal variable)} \\
&= \text{sign}(\alpha^T L K(x) + b) \quad \text{(with dual variable)}
\end{aligned}
$$

Optimization algorithm

- **Solution α* can be computed with the following iterative scheme :**

$$\text{Initialization : } \alpha^{k=0} = \beta^{k=0} = 0_n \in \mathbb{R}^n$$

$$\text{Time steps satisfy } \tau_\alpha \tau_\beta \leq \frac{1}{\|Q\|\|L\|} \text{ s.a. } \tau_\alpha = \frac{1}{\|Q\|}, \tau_\beta = \frac{1}{\|L\|}$$

Iterate :

$$\alpha^{k+1} = \mathrm{P}_{0\leq\cdot\leq\lambda}\big((\tau_\alpha Q + \mathrm{I}_n)^{-1}(\alpha^k + \tau_\alpha Q - \tau_\alpha L \beta^k)\big)$$

$$\beta^{k+1} = \beta^k + \tau_\beta L \alpha^{k+1}$$

$$\text{At convergence, we have : } \alpha^\star$$

$$\text{Classification function : } f_{\mathrm{SVM}}(x) = \mathrm{sign}(\alpha^{\star T} LK(x) + b^\star) \in \pm 1$$

Generalized kernel

Supervised learning for classification

- **SVM belongs to the class of supervised classification algorithms.**
- **In general, algorithms of this class can be described as follows:**

$$\min_{f \in \mathcal{H}_K} \; \|f\|^2_{\mathcal{H}_K} + \lambda \sum_{i=1}^{n} L_{\text{data}}(f_i, \ell_i)$$

with

$$\text{Representer theorem : } \; f(x) = \text{sign}\Big(\sum_{i=1}^{n} \alpha_i K(x, x_i) \Big) \in \pm 1$$

$$\text{Norm of } f \text{ in RKHS : } \|f\|^2_{\mathcal{H}_K} = \langle f, f \rangle_{\mathcal{H}_K} = \sum_{ij} f_i f_j K_{ij} = f^T K f \quad \text{(smoothness/regularity of } f\text{)}$$

$$\|f\|^2_{\mathcal{H}_K} = \|w\|^2_2 \;\; \text{for} \;\; f(x) = w^T x \quad \text{(linear SVM)}$$

$$\text{Misclassification error : } \; L_{\text{data}}(s_i, \ell_i) = L_{\text{Hin}}(d_i = s_i \ell_i) = \max(0, 1 - d_i) \quad \text{(Hinge loss)}$$

Hyper-parameter $\lambda > 0$ controls the trade-off between regularization and data fidelity.

Lab 3 : Kernel/non-linear SVM

- Run code03.ipynb and analyze kernel SVM result on
  - Noisy non-linear data points
  - Real-world text documents

Real-world graph of articles

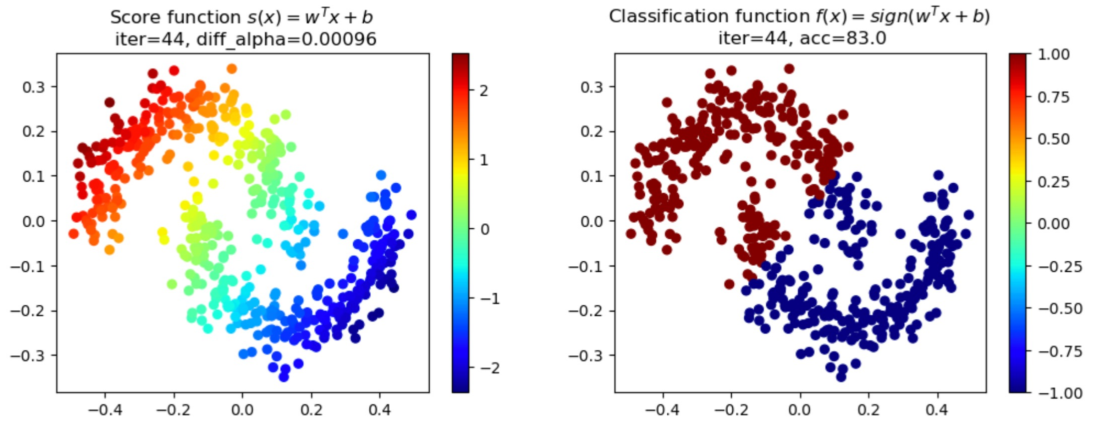
```
# Dataset
mat = scipy.io.loadmat('datasets/data_20news_50labels.mat')
Xtrain = mat['Xtrain']
l_train = mat['l'].squeeze()
n = Xtrain.shape[0]
d = Xtrain.shape[1]
nc = len(np.unique(Cgt_train))
print(n,d,nc)
Xtest = mat['Xtest']
Cgt_test = mat['Cgt_test'] - 1; Cgt_test = Cgt_test.squeeze()
```

```
50 3684 2
```

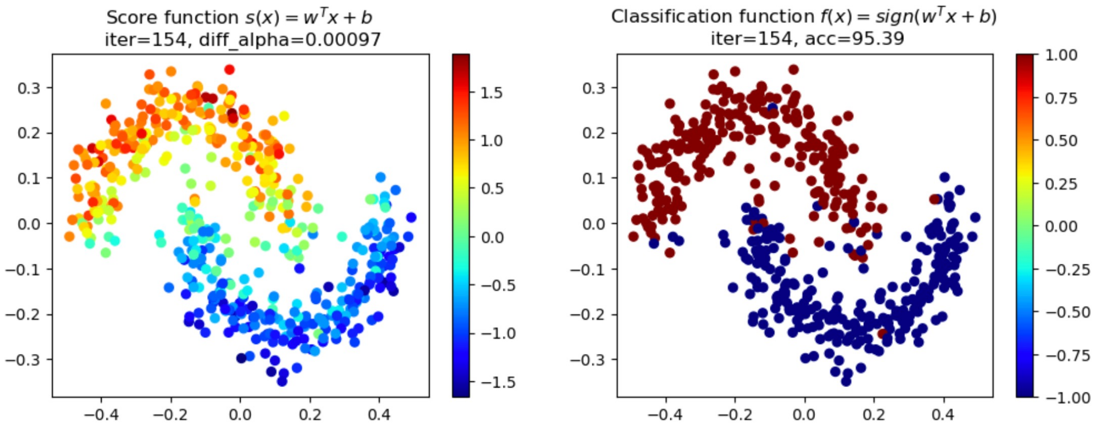
Run kernel SVM

```
# Run kernel SVM

# Compute Gaussian kernel, L, Q
sigma = 0.5; sigma2 = sigma**2
Ddist = sklearn.metrics.pairwise.pairwise_distances(Xtrain, Xtrain, metric='euclidean', n_jobs=1)
Ker = np.exp(- Ddist**2 / sigma2)
Ddist = sklearn.metrics.pairwise.pairwise_distances(Xtrain, Xtest, metric='euclidean', n_jobs=1)
KXtest = np.exp(- Ddist**2 / sigma2)
l = l_train
L = np.diag(l)
Q = L.dot(Ker.dot(L))

# Time steps
tau_alpha = 10/ np.linalg.norm(Q,2)
tau_beta = 0.1/ np.linalg.norm(L,2)

# For conjuguate gradient
Acg = tau_alpha* Q + np.eye(n)

# Pre-compute J.K(Xtest) for test data
LKXtest = L.dot(KXtest)

# Error parameter
lamb = 3 # acc: 87.5
```

```
Kernel SVM  iter, diff_alpha : 100 0.00099
            acc : 87.5
```



Linear SVM



Kernel SVM

Outline

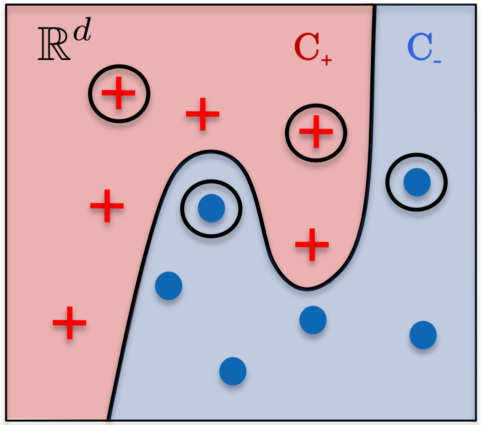
- Supervised classification
- Linear SVM
- Soft-margin SVM
- Kernel techniques
- Non-linear/kernel SVM
- Graph SVM
- Conclusion

Semi-supervised classification

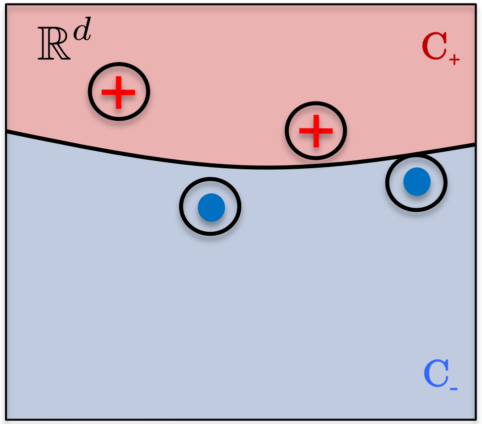
- Semi-supervised classification (SSC) leverages both labeled and unlabeled data to boost the classification process.
- Labeled data, annotated by humans, provide precise insights into class membership, offering a rich information for learning.
- However, human annotation is time-consuming, costly, susceptible to human biases and errors.
- In contrast, unlabeled data depict the underlying structure of the data distribution.
- Collecting unlabeled data is efficient, cheap, but inherently noisy.
- SSC proves particularly beneficial when labeled data are scarce.
  - The situation where $\mathrm{n} \ll \mathrm{m}$, where the number n of labeled instances is significantly smaller than the number m of unlabeled instances.
  - An extreme scenario is when each class has only one labeled instance, $\mathrm{n} = 1$.

Geometric structure

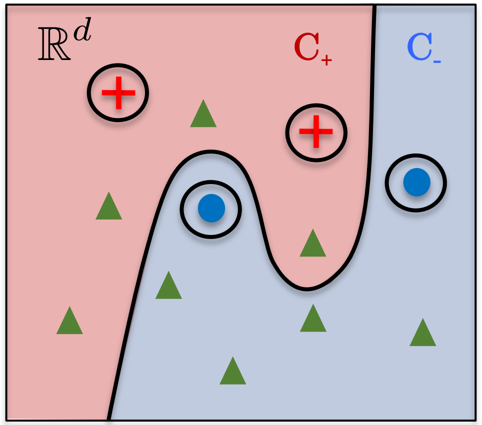
- Unlabeled data encapsulate valuable statistical information, particularly the geometric structure of the data distribution.
- How to leverage this information within the supervised SVM classification framework?



Labeled data +, ●
Supervised classification



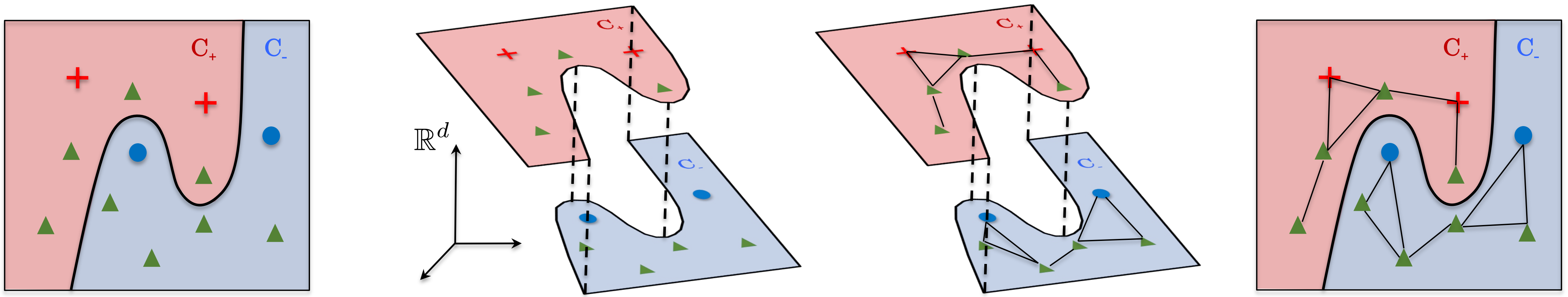
Labeled data +, ●
Supervised classification



Labeled data +, ●
Unlabeled data ▲
Semi-supervised classification

Manifold and graph

- The data distribution remains unchanged regardless of whether labels are present or absent.
- Both labeled and unlabeled data points are assumed to belong to a manifold within the d-dimensional feature space.
- This manifold is estimated using a k-nearest neighbor graph constructed from the data points, serving as an approximation of the underlying manifold structure.



Labeled data +, ●
Unlabeled data ▲
Semi-supervised classification on manifold

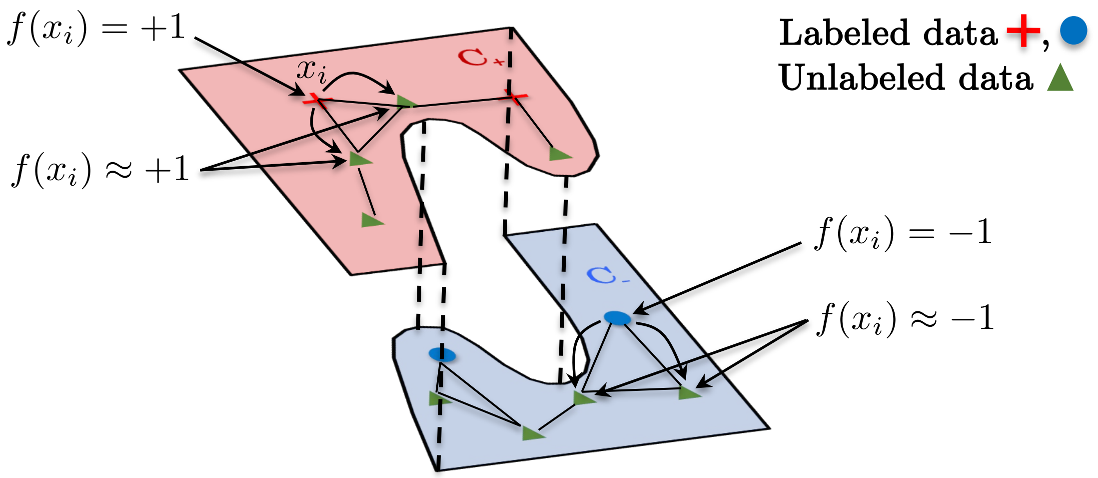
Manifold M embedded in R^d. Data points are sampled from M.

Manifold M is represented by a k-NN graph of the data points.

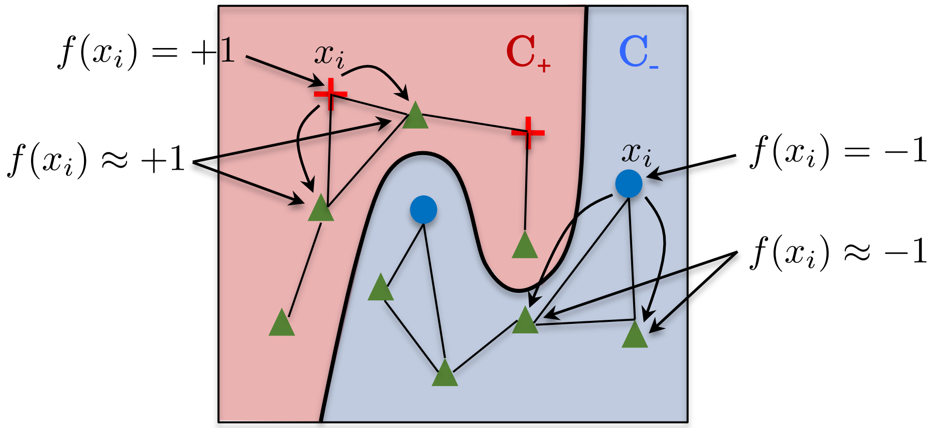
Graph G=(V,E,A) k-NN graph

Manifold regularization

- We aim to ensure that the classification function $f(x)$ exhibits smoothness across the manifold, which is approximated by the k-NN graph.
- This smoothness constraint will propagate the label information throughout the graph, i.e. neighboring data points will tend to share the same label.



Manifold M is represented by a
k-NN graph of the data points.



Graph G=(V,E,A)
k-NN graph

Graph regularization

- Graph regularization is usually implemented through loss minimization techniques.
- A widely used regularization loss is the Dirichlet energy[1], which is defined as:

$$
\begin{aligned}
&\int_{\mathcal{M}} |\nabla f|^2 && \text{(continuous Dirichlet energy)} \\
&\approx \sum_{ij \in V} A_{ij}|f(x_i) - f(x_j)|^2 && \text{(discrete Dirichlet energy)} \\
&\approx f^T L f \in \mathbb{R},\ f \in \mathbb{R}^n, L = \mathrm{I} - D^{-1/2} A D^{-1/2} \in \mathbb{R}^{n\times n} && \text{(Laplacian matrix)} \\
&\qquad D = \mathrm{diag}(d) \in \mathbb{R}^{n\times n}, d = A 1_n \in \mathbb{R}^n && \text{(degree vector)}
\end{aligned}
$$

- Minimizing the Dirichlet energy enforces the smoothness of the function on the graph domain, i.e. $f(x_i) \approx f(x_j)$ for $j \in \mathcal{N}_i$, ensuring that the function values at neighboring data points are similar.

[1] Belkin, Niyogi, Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering, 2001

Semi-supervised classification with graphs

- **SSC optimization problem with graph smoothness :**

$$\min_{f\in\mathcal{H}_K} \|f\|^2_{\mathcal{H}_K} + \lambda \sum_{i=1}^{n} L_{\text{data}}(f_i,\ell_i) + \gamma \int_{\mathcal{M}} |\nabla f|^2$$

- **Graph SVM[1] :**

$$\min_{f\in\mathcal{H}_K} f^T K f + \lambda \sum_{i=1}^{n} L_{\text{Hin}}(f_i,\ell_i) + \gamma f^T L f$$

with

$$\text{Representer theorem}: \ f(x) = \text{sign}\Big(\sum_{i=1}^{n} \alpha_i K(x,x_i) + b\Big) \in \pm 1$$

Misha Belkin

[1] Belkin, Niyogi, Sindhwani,Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, 2006

Optimization algorithm

- **Dual optimization problem :**

$$\min_{f \in \mathcal{H}_K} \; f^T K f + \lambda \sum_{i=1}^{n} L_{\text{Hin}}(f_i, \ell_i) + \gamma f^T L f$$

with

$$\text{Representer theorem : } f(x) = \text{sign}\Big(\sum_{i=1}^{n} \xi_i^\star K(x, x_i) + b \Big) \in \pm 1$$

$$\text{Optimization problem : } \alpha^\star = \arg \min_{0 \leq \alpha \leq \lambda} \; \frac{1}{2} \alpha^T Q \alpha - \alpha^T 1_n \;\; \text{s.t. } \alpha^T \ell = 0$$

$$\text{with } Q = LHK(\mathrm{I} + \gamma LK)^{-1} HL \in \mathbb{R}^{n \times n}$$

$$\text{Solution : } \xi^\star = (\mathrm{I} + \gamma LK)^{-1} HL \alpha^\star$$

Lab 4 : Graph SVM

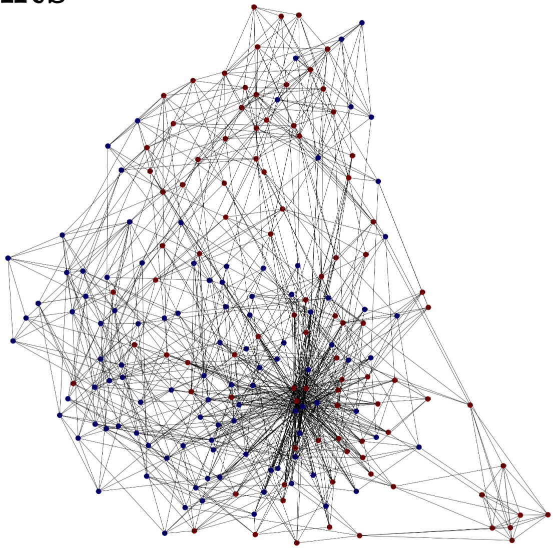
- Run code04.ipynb and analyze Graph SVM result on
  - Noisy non-linear data points
  - Real-world text documents

Real-world graph of articles

Dataset has 10 labeled data and 40 unlabeled data

```
# Dataset
mat = scipy.io.loadmat('datasets/data_20news_10labels_40unlabels.mat')
Xtrain = mat['Xtrain']
n = Xtrain.shape[0]
l_train = mat['l'].squeeze()
d = Xtrain.shape[1]
Xtest = mat['Xtest']
Cgt_test = mat['Cgt_test'] - 1; Cgt_test = Cgt_test.squeeze()
nc = len(np.unique(Cgt_test))
print(n,d,nc)
num_labels = np.sum(np.abs(l_train)>0.0)
print('l_train:',l_train)
print('number of labeled data per class:',num_labels//2)
print('number of unlabeled data:',n-num_labels)
```

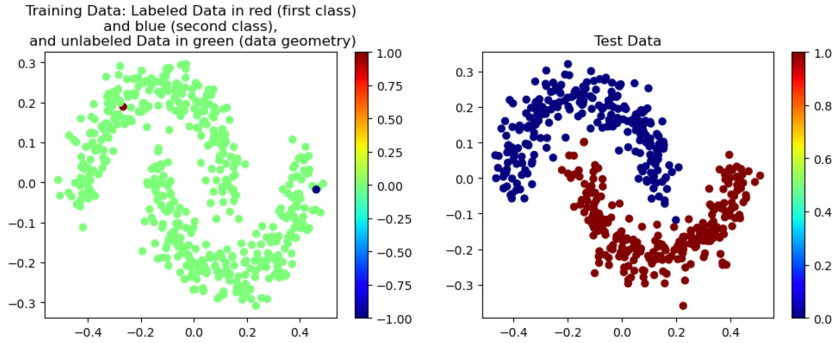
```
50 3684 2
l_train: [-1  0 -1 -1 -1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0 -1  0  0  0  0  0  0  1
  1  0]
number of labeled data per class: 5
number of unlabeled data: 40
```

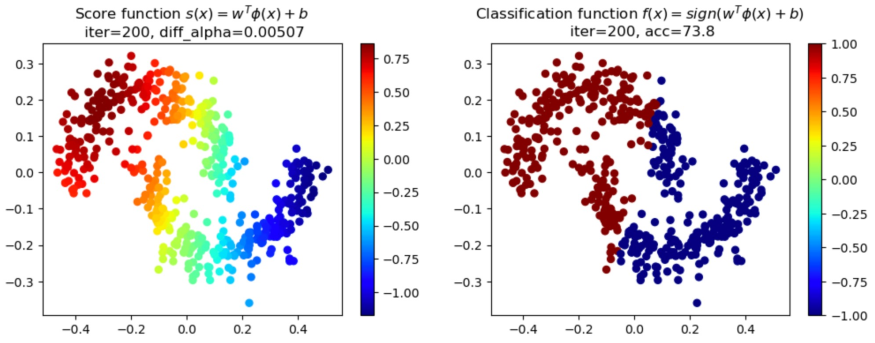


Run Graph SVM

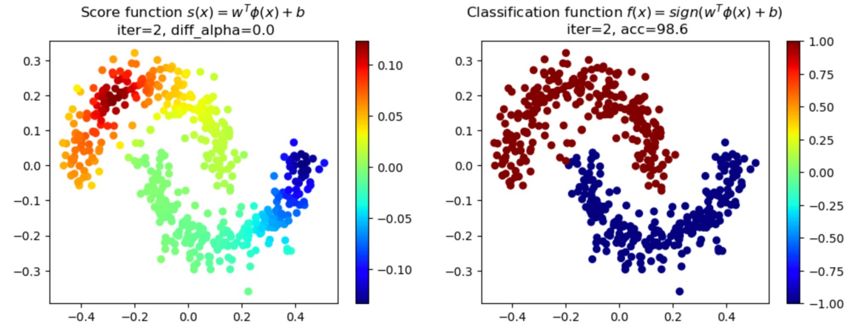
```
# Run Graph SVM

# Compute Gaussian kernel
sigma = 0.5; sigma2 = sigma**2
Ddist = sklearn.metrics.pairwise.pairwise_distances(Xtrain, Xtrain, metric='euclidean', n_jobs=1)
Ker = np.exp(- Ddist**2 / sigma2)
Ddist = sklearn.metrics.pairwise.pairwise_distances(Xtrain, Xtest, metric='euclidean', n_jobs=1)
KXtest = np.exp(- Ddist**2 / sigma2)

# Compute kNN graph
kNN = 8
gamma = 100
A = construct_knn_graph(Xtrain, kNN, 'cosine')
Lap = graph_laplacian(A).todense()

# Compute Indicator function of labels
H = np.zeros([n])
H[np.abs(l_train)>0.0] = 1
H = np.diag(H)
```

```
k-NN graph with cosine distance
Graph SVM  iter, diff_alpha : 2 0.0
        acc : 78.5
```





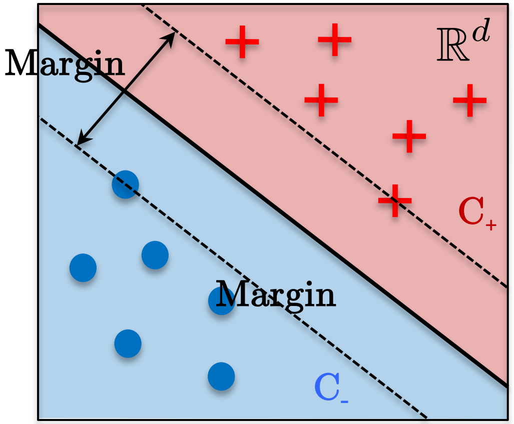
Kernel SVM



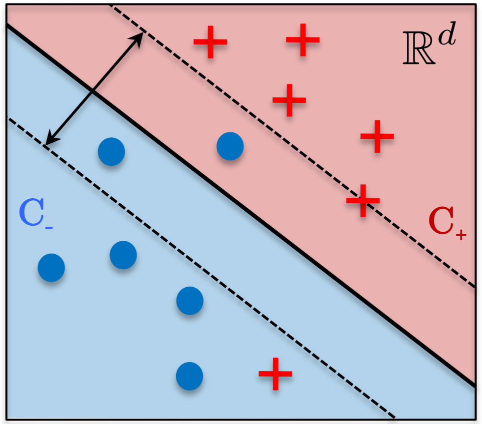
Graph SVM

Outline

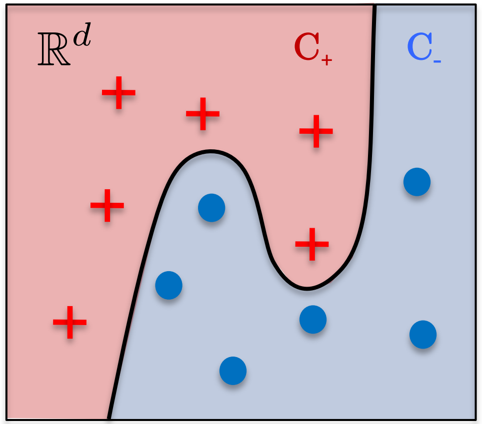
History of SVM techniques



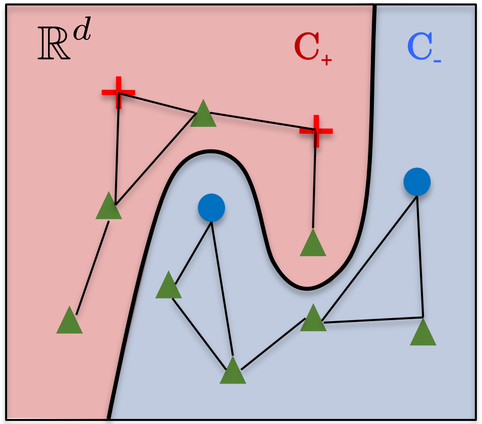
Linear SVM[1]
Supervised learning



Soft-Margin SVM[2]
Supervised learning



Non-Linear/Kernel SVM[3]
Supervised learning



Graph SVM[4]
Semi-supervised learning

[1] Vapnik, Chervonenkis, On a perceptron class, 1964
[2] Cortes, Vapnik, Support-vector networks, 1995
[3] Boser, Guyon, Vapnik, A training algorithm for optimal margin classifiers, 1992
[4] Belkin, Niyogi, Sindhwani,Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, 2006

Summary

- **General class of semi-supervised optimization techniques :**

$$\min_{f \in \mathcal{H}_K} \; \|f\|^2_{\mathcal{H}_K} + \lambda \sum_{i=1}^{n} L_{\text{data}}(f_i, \ell_i) + \gamma L_{\text{graph}}(f)$$

where

Norm of f in RKHS : $\|f\|^2_{\mathcal{H}_K} = f^T K f$ (smoothness/regularity of f)

Misclassification error : $L_{\text{data}}(f_i, \ell_i)$ (training prediction)

Graph regularization : $L_{\text{graph}}(f)$ (smoothness of f on graph domain)

with

$$L_{\text{data}} = \begin{cases} \text{Hinge} \\ L_2 \\ L_1 \\ \text{Huber} \\ \text{Logistic} \end{cases} \quad \text{and} \quad L_{\text{graph}} = \begin{cases} \text{Dirichlet :} & \|\nabla \cdot\|^2 \\ \text{Total variation}^{[1]} : & \|\nabla \cdot\|_1 \\ \text{Wavelets :} & \|\nabla_{\text{wav}} \cdot\|^2 \end{cases}$$

[1] Bresson, Zhang, TV-SVM: Total variation support vector machine for semi-supervised data classification, 2012

Questions?