

CS5284 : Graph Machine Learning

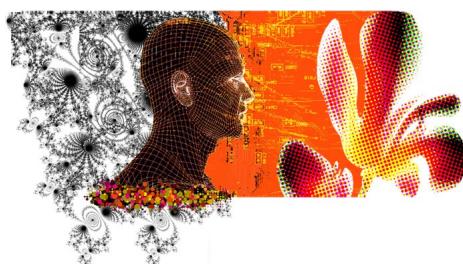
Lecture 4 : Graph SVM

Semester 1 2025/26

Xavier Bresson

<https://x.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 on graphs
 - Graph-based visualization
- 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
 - Graph generation & molecular science
 - Material design
 - Integrating GNNs and LLMs
 - Deep Learning for Combinatorial optimization

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 Sept 2024, 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, e.g. first training step of ChatGPT.

Step 1) Graph captures the distribution of the unlabeled datapoints \rightarrow unsupervised

Step 2) Train SVM \rightarrow supervised

[semi-supervised learning]

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.
via kernels
 - 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

- 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].



Vladimir Vapnik

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

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

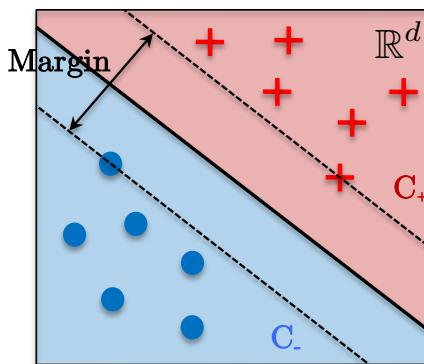
$\ell_i \in \{-1, 1\}$ (data label)

Positive label :

$$x_i, \ell_i = +1$$

Classification function :

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



Negative label :

$$x_i, \ell_i = -1$$

Classification function :

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

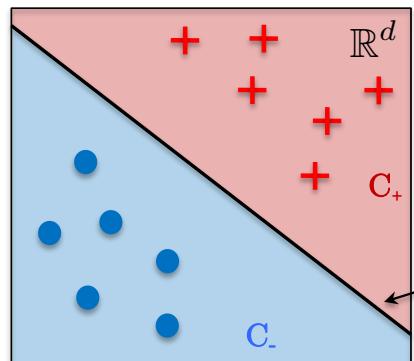


[1] Vapnik, Chervonenkis, On a perceptron class, 1964

Linear SVM

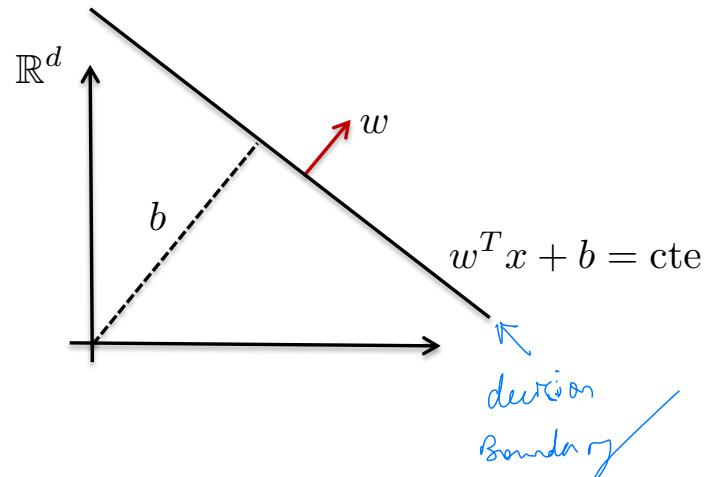
- Assumption^[1] : Training and test datasets are **linearly separable**, i.e. data can be separated with a straight line in 2D, a plane in 3D and a hyper-plane in higher dimensions.
- A hyper-plane is parameterized with two variables (w, b) , where w is the normal vector of the hyper-plane, i.e. determining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

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



Hyperplane
separator between
the two classes

orthogonal to
decision boundary

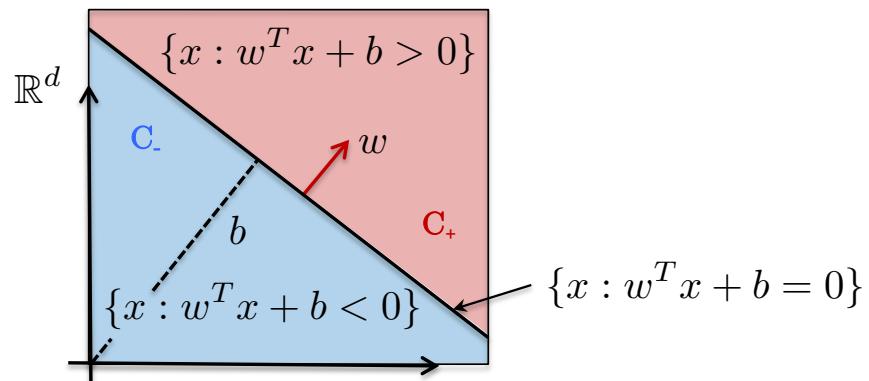


[1] Vapnik, Chervonenkis, On a perceptron class, 1964

SVM classifier

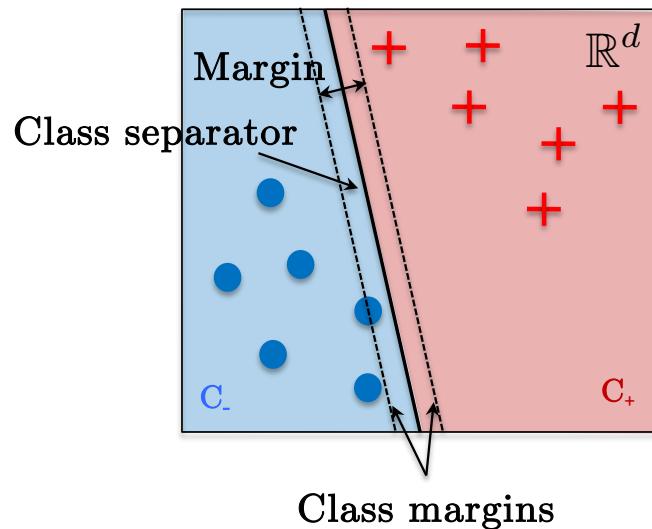
- Classification function :

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



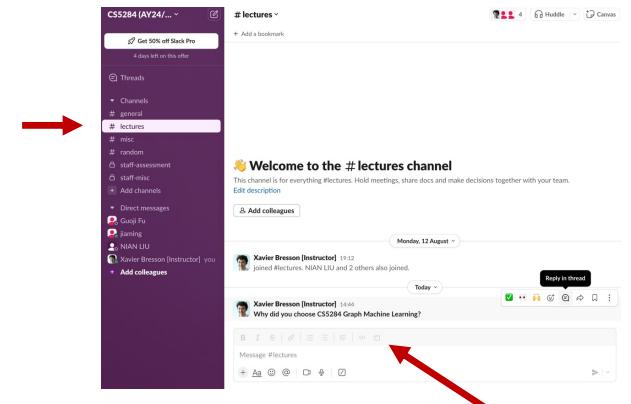
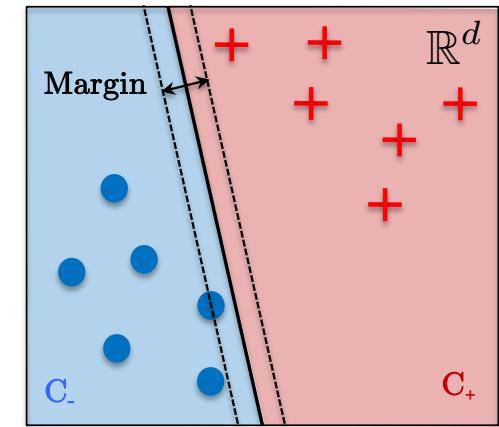
Maximizing class margin

- Hyper-plane $w^T x + b = 0$ is the class separator.
- Hyper-planes $w^T x + b = \pm 1$ are the class margins.



In-lecture question

- How many class separators exist for linearly separable distributions? Why is it important to maximize the margin between the two classes? Please justify.
- In Slack #lectures
 - Identify the question and Reply in thread with a short response
- Answer :



Maximizing class margin

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

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

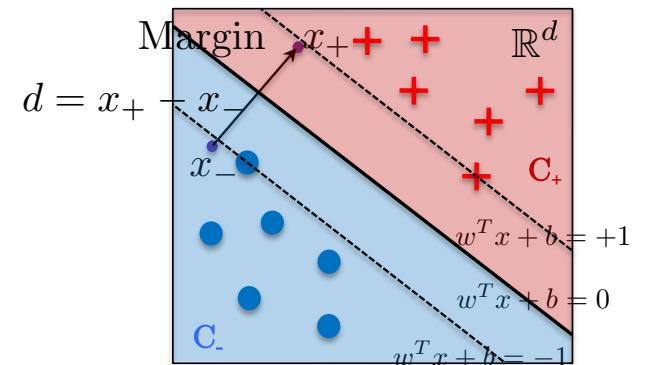
Given that $w^T x_+ + b = +1$ and $w^T x_- + b = -1$ and

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

Then taking the norm : $\|w\|_2 \cdot \|d\|_2 = 2 \Rightarrow \|d\|_2 = \frac{2}{\|w\|_2}$

Finally, $\max_d \|d\|_2 = \frac{2}{\|w\|_2} \Leftrightarrow \min_w \|w\|_2^2$ s.t. $\begin{cases} w^T x_i + b \geq +1 & \text{if } x_i \in C_+ \\ w^T x_i + b \leq -1 & \text{if } x_i \in C_- \end{cases}$

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



$$\text{WTS: } \max_{w,b} \text{margin distance} \equiv \min_{w} \|w\|_2$$

Define a point z_+ on the margin $w^T z_+ + b = 1 \quad \text{--- ①}$

" " " z_- on the margin $w^T z_- + b = -1 \quad \text{--- ②}$

we select z_+ & z_- s.t. $(z_+ - z_-)$ is parallel with w

\therefore The margin distance, $d = \|z_+ - z_-\|_2$

$$\textcircled{1} - \textcircled{2}: w^T(z_+ - z_-) = 2$$

$$\|w\|_2 \cdot \|z_+ - z_-\|_2 = 2, \text{ where } \|\cdot\|_2 \text{ is the L2 norm}$$

$$\Rightarrow \|z_+ - z_-\|_2 = \frac{2}{\|w\|_2}$$

$$\therefore \max_{w,b} \|z_+ - z_-\|_2 \equiv \max_{w,b} \frac{2}{\|w\|_2} \equiv \min_{w,b} \|w\|_2$$

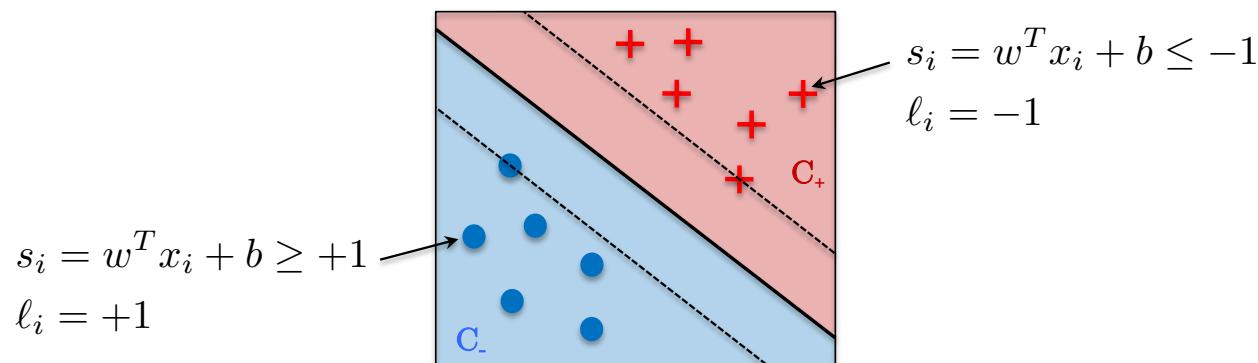
Primal optimization problem

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

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

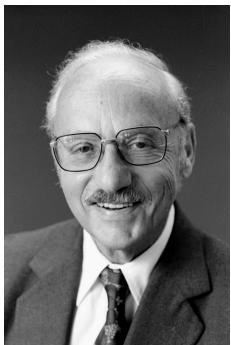
$$\text{with } s_i = w^T x_i + b = \begin{cases} \geq +1 & \text{for } x \in C_+ \\ \leq -1 & \text{for } x \in C_- \end{cases} \quad \text{and} \quad \ell_i = \begin{cases} +1 & \text{if } x \in C_+ \\ -1 & \text{if } x \in C_- \end{cases}$$

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



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.

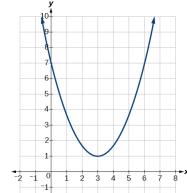


George Dantzig
1914-2005

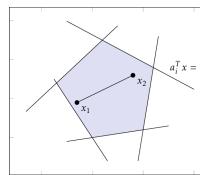
Quadratic Programming

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

Quadratic function



Convex set (polytope)



SVM classifier

$$\text{sign}(c) = \begin{cases} 1 & \text{if } c > 0 \\ 0 & \text{if } c = 0 \\ -1 & \text{otherwise} \end{cases}$$

- [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

Finding b^* after getting w^* via minimization QP in prev slide

Support vectors

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

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

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

which gives

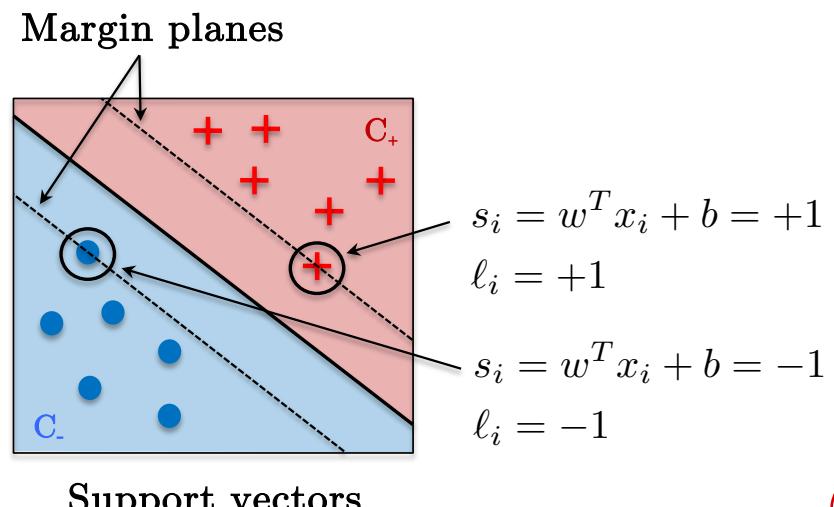
$$b^* = \ell_i - (w^*)^T x_i^{\text{sv}}$$

and on expectation

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

Average b^*

Qn: But how do we determine x_i^{sv} before determining what b^* even is in the first place



Support vectors

If we look at the values of the Lagrangian multipliers in the dual problem (later) after solving the optimization problem,

$\alpha_i > 0 \rightarrow$ constraint $i \rightarrow \ell_i s_i = 1 \rightarrow x_i \text{ is a support vector}$

Dual variable

To introduce the Kernel matrix later

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

Trainable weights

Given $w = \sum_i \alpha_i \ell_i x_i \in \mathbb{R}^d, \alpha_i \in \mathbb{R}$

See eqn ① in proof below

linear for now just to
re-represent the previous
primal problem!

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

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

$$\text{Note: } \frac{1}{2} \|w\|_2^2 = \frac{1}{2} w^T w$$

Primal problem

$$\min_{w,b} \underbrace{\frac{1}{2} w^T w}_{f(w,b)}$$

$$\text{s.t. } \underbrace{l_i(w^T \phi(x_i) + b) - 1 \geq 0}_{h_i(w,b)} \quad \forall i \in \{1, \dots, n\}$$

n constraints

Dual problem

$$\max_{\alpha_1, \dots, \alpha_n} \min_{w,b} \underbrace{\frac{1}{2} w^T w - \sum_{i=1}^n \alpha_i [l_i(w^T \phi(x_i) + b) - 1]}_{\text{Lagrange function, } L(w, b, \alpha)}$$

s.t. $\alpha_i \geq 0 \quad \forall i = 1, \dots, n$

$$\frac{\partial L}{\partial w} = w - \left[\sum_{i=1}^n \alpha_i l_i \phi(x_i) \right] = 0 \Rightarrow w^* = \sum_{i=1}^n \alpha_i l_i \phi(x_i) \quad \text{--- ①}$$

$$\frac{\partial L}{\partial b} = - \sum_{i=1}^n \alpha_i l_i = 0 \Rightarrow \underbrace{\sum_{i=1}^n \alpha_i l_i}_{\therefore \alpha^T l = 0} = 0 \quad \text{--- ②}$$

where $l = [l_1, \dots, l_n]^T$

$$\text{Optional: KKT complementary slackness: } \forall i = 1, \dots, n \quad \alpha_i (l_i (w^T \phi(x_i) + b) - 1) = 0 \quad \text{--- ③}$$

$$L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^n \alpha_i [l_i (w^T \phi(x_i) + b) - 1]$$

$$\frac{1}{2} w^T w = \frac{1}{2} \left[\sum_{i=1}^n \alpha_i l_i \phi(x_i) \right]^T \left[\sum_{j=1}^n \alpha_j l_j \phi(x_j) \right] \quad (\text{From } ①)$$

$$= \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j l_i l_j \phi(x_i)^T \phi(x_j)$$

↓↓↓
scalars

$$\sum_{i=1}^n \alpha_i [l_i (w^T \phi(x_i) + b) - 1] = \sum_{i=1}^n \alpha_i l_i w^T \phi(x_i) + \alpha_i l_i b - \alpha_i$$

$$= \sum_{i=1}^n \alpha_i l_i w^T \phi(x_i) + \underbrace{\left(\sum_{i=1}^n \alpha_i l_i \right) b}_{=0 \text{ from } ②} - \sum_{i=1}^n \alpha_i$$

$$= \sum_{i=1}^n \alpha_i l_i \left[\sum_{j=1}^n \alpha_j l_j \phi(x_j) \right]^T \phi(x_i) - \sum_{i=1}^n \alpha_i \quad (\text{From } ①)$$

$$= \sum_{i,j=1}^n \alpha_i \alpha_j l_i l_j \phi(x_i)^T \phi(x_j) - \sum_{i=1}^n \alpha_i$$

$$\therefore L(w, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j l_i l_j \phi(x_i)^T \phi(x_j)$$

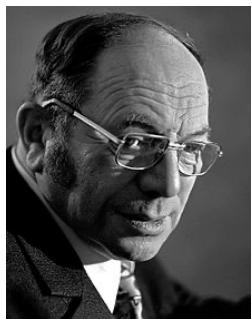
$$= \alpha^T \mathbf{1}_n - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j l_i l_j K_{ij}$$

$$= \alpha^T \mathbf{1}_n - \frac{1}{2} \alpha^T L K L \alpha$$

$$= \alpha^T \mathbf{1}_n - \frac{1}{2} \alpha^T Q \alpha \quad \text{where } Q = L K L$$

Dual optimization problem

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



Leonid Kantorovich
1912-1986

$\ell_i \geq 0$
 $\ell_i = \langle \cdot, \cdot \rangle$

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

is equivalent to

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

with $Q = LKL \in \mathbb{R}^{n \times n}$

$$L = \text{diag}(\ell) \in \mathbb{R}^{n \times n} \quad \leftarrow \text{label matrix}$$

$$\ell = (\ell_1, \dots, \ell_n) \in \mathbb{R}^n$$

$$K \in \mathbb{R}^{n \times n}, K_{ij} = x_i^T x_j \in \mathbb{R} \quad (\text{linear kernel})$$

see proof or prev slide
eqn ② above in proof

- [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 :

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

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

Iterate :

$$\alpha^{k+1} = P_{\geq 0}((\tau_\alpha Q + I_n)^{-1}(\alpha^k + \tau_\alpha 1_n - \tau_\alpha L \beta^k)) \in \mathbb{R}^n$$

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

At convergence, we have : α^*

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



Narendra Karmarkar

Karmarkar's
algorithm (?)

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

Projection on a convex set

Theorem 11.8 (Projection theorem)

Let C be a closed convex set in \mathbf{R}^n .

(a) For every $z \in \mathbf{R}^n$, there exists a unique minimizer (denoted as $\Pi_C(z)$) and called as the projection of z onto C) of

$$\min \left\{ \frac{1}{2} \|x - z\|^2 \mid x \in C \right\}$$

dist between x & z , $x \in C$

where $\|\cdot\|$ is the Euclidean norm.

(b) $x^* := \Pi_C(z)$ is the projection of z onto C if and only if

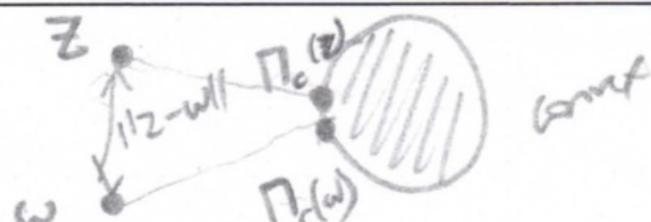
$$\langle z - x^*, x - x^* \rangle \leq 0 \quad \forall x \in C.$$

(c) For any $z, w \in \mathbf{R}^n$,

$$\|z - x^*\| \|x - x^*\| \cos \theta \leq 0 \rightarrow 90^\circ \leq \theta \leq 270^\circ$$

$$\|\Pi_C(z) - \Pi_C(w)\| \leq \|z - w\|.$$

only for convex sets



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, 0
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(0,2)
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
LXtest = L.dot(Xtrain.dot(Xtest.T))

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

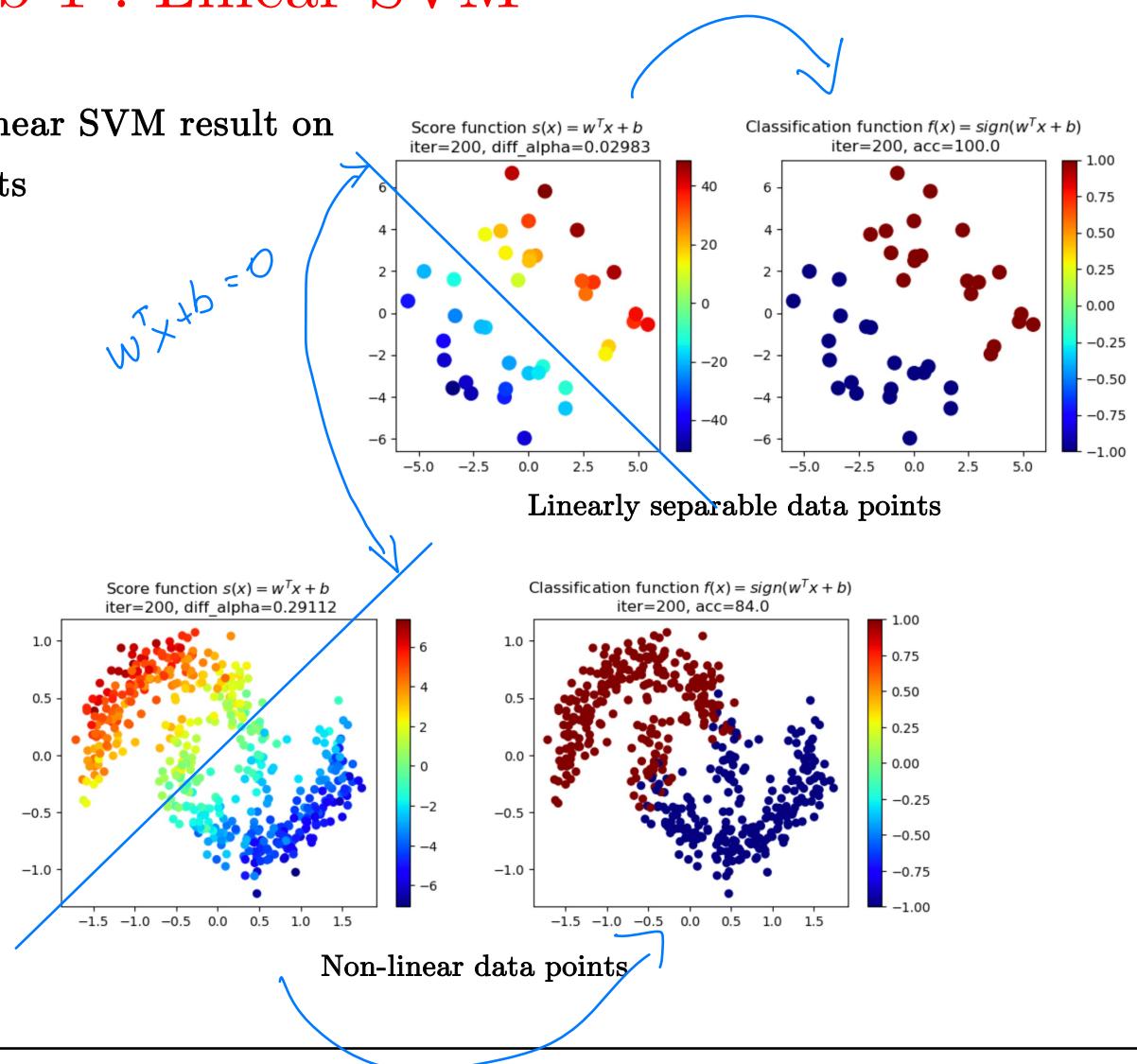
# Loop
k = 0
diff_alpha = 1e6
num_iter = 101
while (diff_alpha>1e-3) & (k<num_iter):

    # Update iteration
    k += 1

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

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

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



Outline

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

Noise

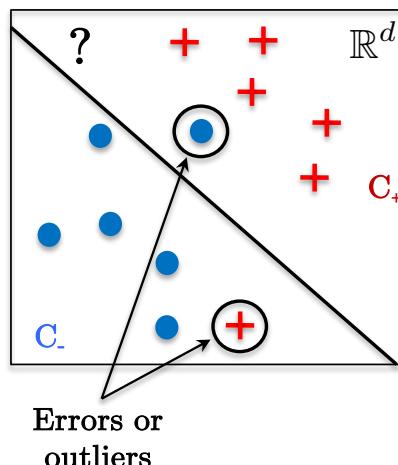
- Real-world data often contains noise and outliers, which do not satisfy the assumption of linearly separable data points.
- When dealing with non-linearly separable data, there is no mathematical solution for standard or hard-margin SVM because there does not exist a linear separator that can split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

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

\Downarrow

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

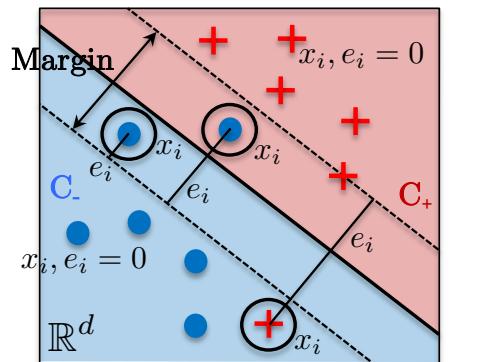
(Standard SVM)

Hard clustering

(Soft-margin SVM)

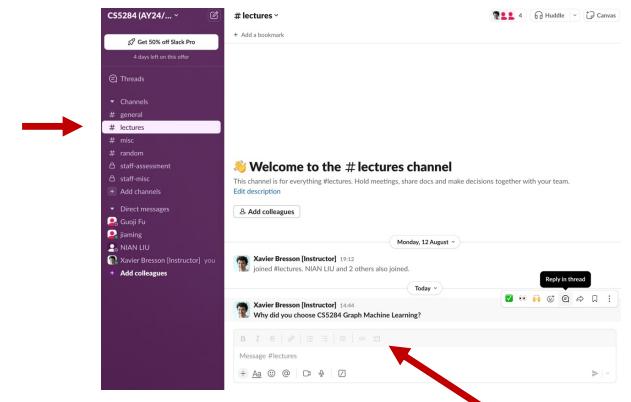
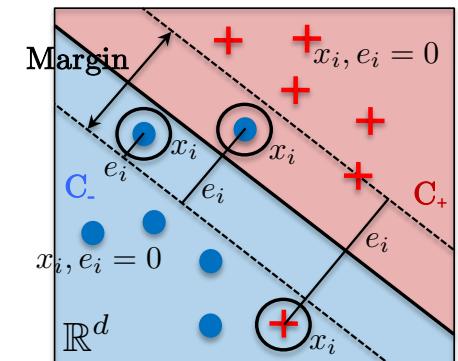
Xo balance between error and width

Trade-off between large margin and small errors



In-lecture question

- What is the impact of varying the regularization parameter λ ? Specifically, how do small versus large values of λ affect the model? Please justify.
- In Slack #lectures
 - Identify the question and Reply in thread with a short response
- Answer :



- 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 :

$$\min_{w,e} \|w\|_2^2 + \lambda \sum_{i=1}^n e_i \quad \text{s.t.}$$

$$\begin{cases} w^T x_i + b \geq +1 - e_i & \text{for } x_i \in C_+ \\ w^T x_i + b \leq -1 + e_i & \text{for } x_i \in C_- \\ e_i \geq 0 & \text{for } x_i \in V \end{cases}$$

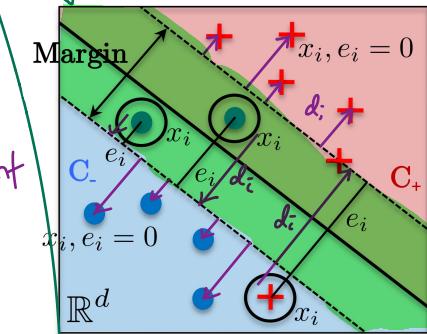
$$\Leftrightarrow \min_{w,e} \|w\|_2^2 + \lambda \sum_{i=1}^n \max(0, 1 - \ell_i s_i),$$

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

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

Hinge loss, $L_{\text{Hin}}(d)$

displacement (ie distance with direct²) btwn datapoint and margin



$$L_{\text{Hin}}(d) = \max(0, 1 - d)$$

constraints

(1)

(2)

\star

SVM

Hinge loss

$d < 1$

$L_{\text{Hin}}(d) > 0$

Misclassified

$d \geq 1$

$L_{\text{Hin}}(d) = 0$

Correctly classified

Loss functions

- There exist multiple loss functions^[1] :

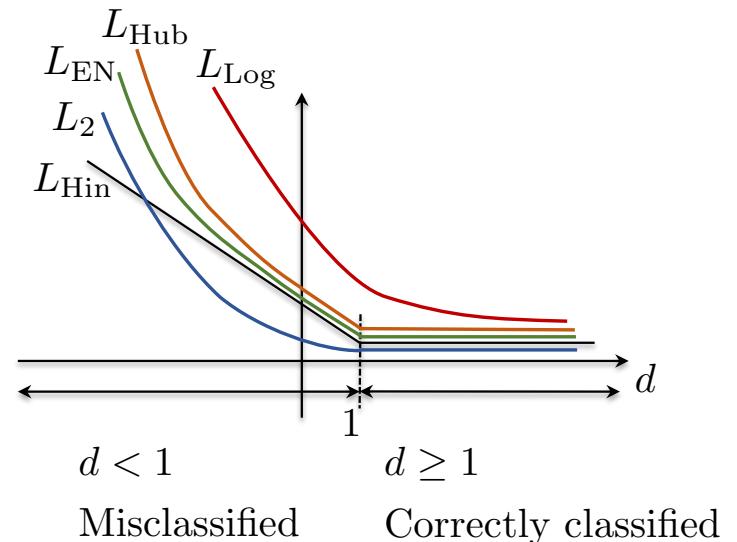
$$L_2(d_i) = \begin{cases} (1 - d_i)^2 & \text{if } d_i < 1 \\ 0 & \text{otherwise} \end{cases} \quad (\text{L2 loss})$$

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

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

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

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



Not really used in practice :: Cross-Entropy works really well
 But still used in research for its nice geometric properties & interpretation

[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 :

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

is equivalent to

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

Modification

$$\text{with } Q = LKL \in \mathbb{R}^{n \times n}$$

$$L = \text{diag}(\ell) \in \mathbb{R}^{n \times n}$$

$$\ell = (\ell_1, \dots, \ell_n) \in \mathbb{R}^n$$

$$K \in \mathbb{R}^{n \times n}, K_{ij} = x_i^T x_j \in \mathbb{R} \quad (\text{linear kernel})$$

Optimization algorithm

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

Initialization : $\alpha^{k=0} = 0_n \in \mathbb{R}^n$ and $\beta^{k=0} = 1_n \in \mathbb{R}^n$

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

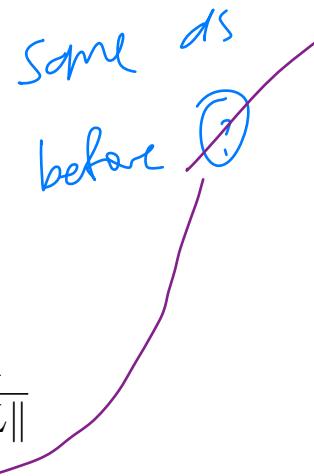
Iterate :

$$\alpha^{k+1} = P_{0 \leq \cdot \leq \lambda}((\tau_\alpha Q + I_n)^{-1}(\alpha^k + \tau_\alpha 1_n - \tau_\alpha L \beta^k))$$

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

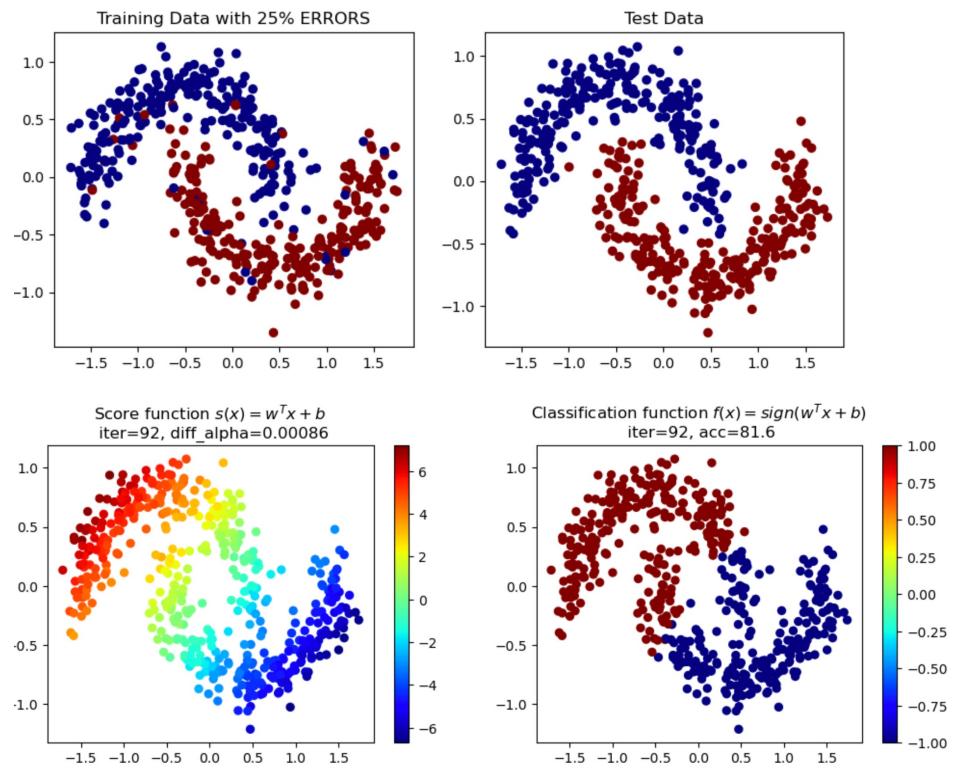
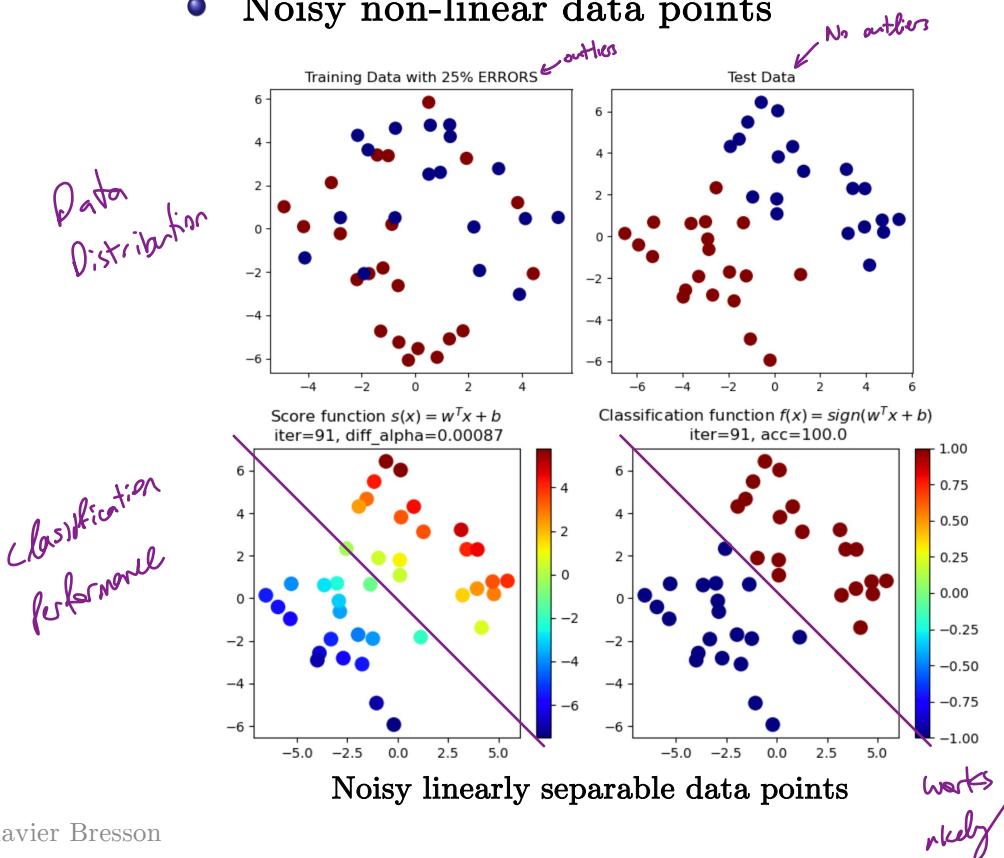
At convergence, we have : α^*

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



Lab 2 : Soft-margin SVM

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



Outline

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

Approximate a function using a linear combination of kernels (we apply some kernel which are centered on training datapoints)

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 $K(x, x) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_+$ that can reproduce any smooth function $h(x) : \mathbb{R}^d \rightarrow \mathbb{R}$.
- 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 :

$\xrightarrow{\text{maps data to RKHS space, } \mathcal{H}}$

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

$\xrightarrow{\text{inner product}}$

$(x, x') \in \mathbb{R}^d \times \mathbb{R}^d \xrightarrow{\phi} (\phi(x), \phi(x')) \in \mathcal{H} \times \mathcal{H} \xrightarrow{\langle \cdot, \cdot \rangle_{\mathcal{H}}} K(x, x') \in \mathbb{R}$

$\xrightarrow{\text{scalar}}$

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

$\xrightarrow{=} h(x) \in \mathbb{R}$

$\text{Note: } h(\cdot) \in \mathcal{H}$

$K(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$

directly skipping: Kernel trick

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



David Hilbert
1862-1943

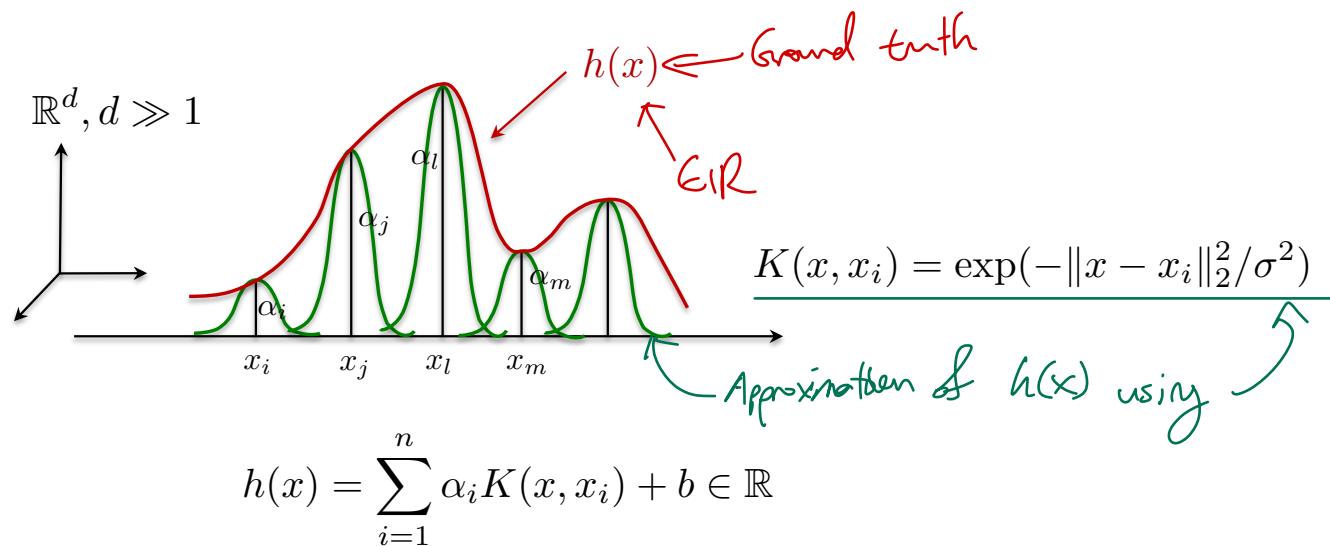


Bernhard
Schölkopf

intractable to compute $\xrightarrow{\phi(\cdot) : \mathbb{R}^d \rightarrow \mathcal{H}}$
 $h(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}$ & $h(\cdot) \in \mathcal{H}$
 $h(x) \in \mathbb{R}$ ← scalar

Representer Theorem

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



with the most common kernels are defined as

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

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

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

Hypersparameters (Used to be chosen via cross-validation,
but not anymore : too expensive)

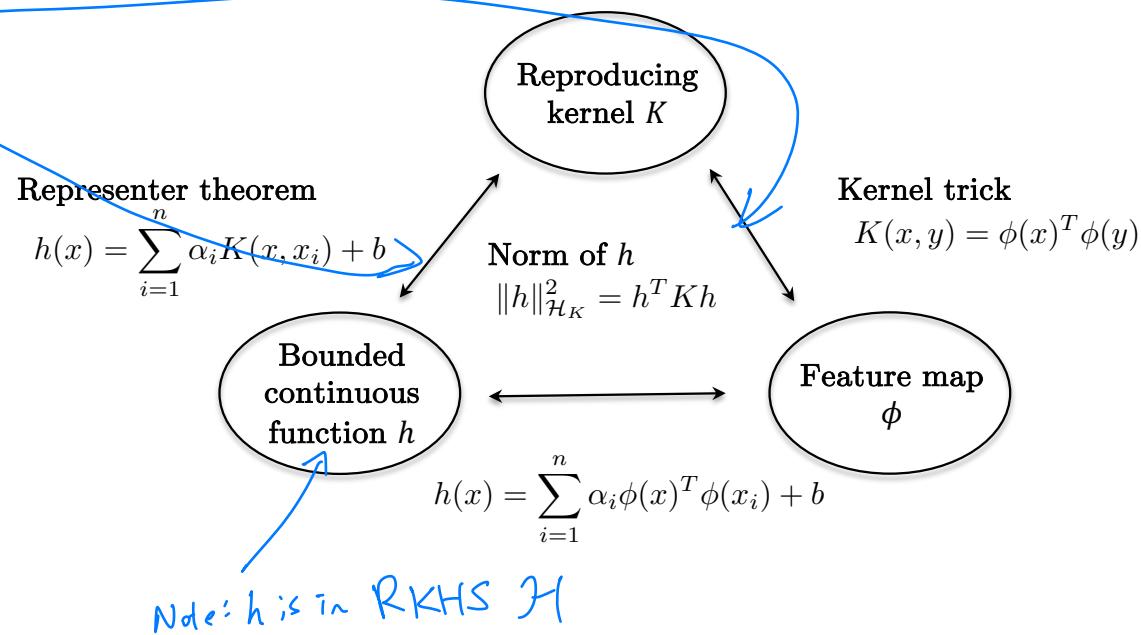
① Then what do we use now

ϕ : Just a mapping function $X \rightarrow \phi(x)$
 Kernel K : $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

Feature map, kernel trick and interpolation

- Any feature map ϕ defines a reproducing kernel K , and inversely. ie: $\forall K, \exists \phi$ s.t. $\phi(x)^T \phi(y) = K(x, y)$
- Any kernel K can be used to design a smooth high-dim function h .

$\forall K, \exists \phi$ s.t. $\phi(x)^T \phi(y) = K(x, y)$
 ie. $\forall K, K$ can design some h .
 the fuck ?

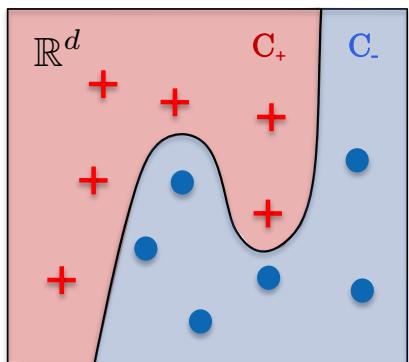


Outline

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

Feature engineering for non-linear data

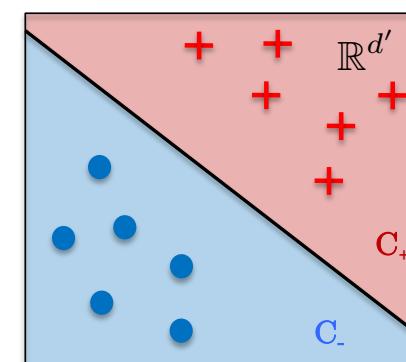
- Linear models, s.a. original and soft-margin SVM, assume linearly separable data points.
- But in many real-world scenarios, datasets are not linearly separable, i.e. a hyper-plane cannot distinguish between distinct classes.
- How to address this challenge and classify complex/non-linear datasets 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

Feature map $\phi(\cdot)$

$$x \in \mathbb{R}^d \rightarrow \phi(x) \in \mathbb{R}^{d'} \\ d' > d \text{ (possibly } d' \gg d\text{)}$$



Linear dataset
and separator

via ϕ , but we don't explicitly compute $\phi(\cdot)$ and $\phi(x_i) \forall x_i$

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

Some slide as per lecture

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 $O(d')$ where $d' \gg d$.
- 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 $K = \phi^T \phi$, rather than ϕ individually, making the exact expression of ϕ irrelevant.
 - Some standard kernel operators include :

$$\begin{array}{ll} K(x_i, x_j) = x_i^T x_j & \text{Time consuming} \\ K(x_i, x_j) = \phi(x_i)^T \phi(x_j) = \exp(-\|x_i - x_j\|_2^2 / \sigma^2) & \text{(linear kernel for linear k-means)} \\ K(x_i, x_j) = (ax_i^T x_j + b)^c & \text{(Gaussian kernel)} \\ & \text{Efficient kernel computation} \\ & \text{(Polynomial kernel)} \end{array}$$

[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 :

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

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

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

with $Q = LKL \in \mathbb{R}^{n \times n}$

$$L = \text{diag}(\ell) \in \mathbb{R}^{n \times n}$$

$$\ell = (\ell_1, \dots, \ell_n) \in \mathbb{R}^n$$

$$K \in \mathbb{R}^{n \times n}, K_{ij} = \phi(x_i)^T \phi(x_j) \in \mathbb{R} \quad (\text{generalized kernel})$$

Function ϕ is never used explicitly.

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

Non-linear/kernel SVM

- Decision function $f(x)$:

Given $w = \sum_i \alpha_i \ell_i \phi(x_i) \in \mathbb{R}^d$

we have $w^T x = \sum_i \alpha_i \ell_i \phi(x_i)^T \phi(x) \in \mathbb{R}$

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

what is $K(x)$?

\mathbb{R}^n vector

of $K(x_i, x)$

$\begin{bmatrix} K(x_1, x) \\ \vdots \\ K(x_n, x) \end{bmatrix}$

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

Optimization algorithm

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

Initialization : $\alpha^{k=0} = 0_n \in \mathbb{R}^n$ and $\beta^{k=0} = 1_n \in \mathbb{R}^n$

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

Iterate :

$$\alpha^{k+1} = P_{0 \leq \cdot \leq \lambda} \left(\underbrace{(\tau_\alpha Q + I_n)^{-1}}_{\text{conjugate gradient}} (\alpha^k + \tau_\alpha 1_n - \tau_\alpha L \beta^k) \right)$$

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

At convergence, we have : α^*

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

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:

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

Loss
(Can be Hinge / Cross-Ent.,
L2, Elastic L2, etc.)

with

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

① What is the fuck

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

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

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

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))

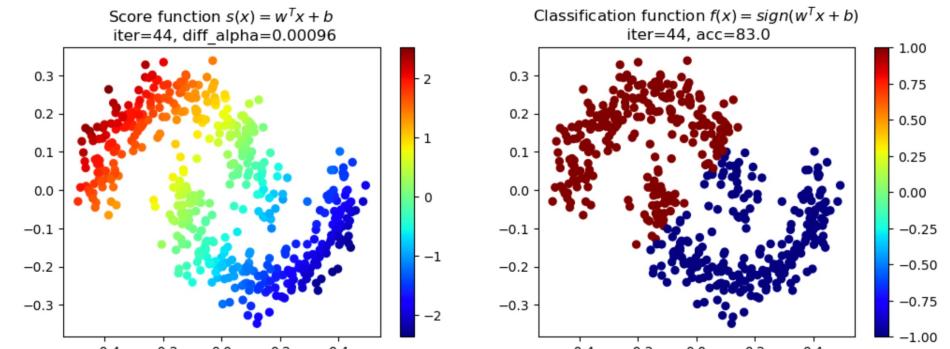
# Time steps
tau_alpha = 10/ np.linalg.norm(Q, 2)
tau_beta = 0.1/ np.linalg.norm(L, 2)

# For conjugate gradient
Acp = tau_alpha* Q + np.eye(n)

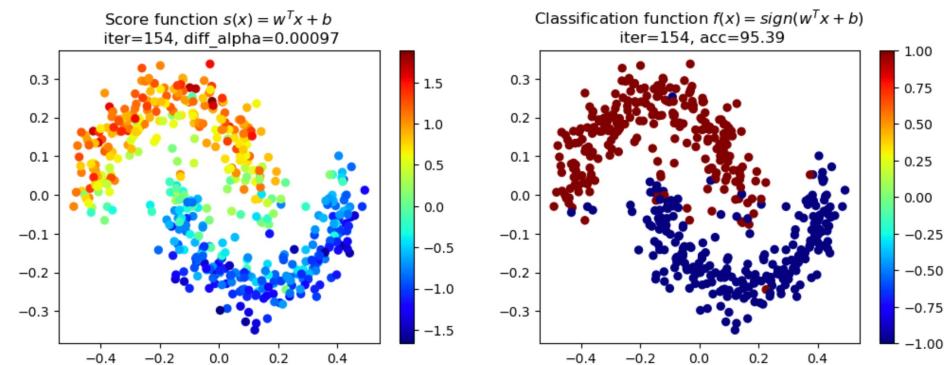
# Pre-compute J.K(Xtest) for test data
LKXtest = L.dot(KXtest)

# Error parameter
lamb = 3 # acc: 87.5
Kernel SVM iter, diff_alpha : 100 0.00099
acc : 87.5
```

Xavier Bresson



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. *but abundance of unlabeled data*
 - The situation where $n \ll m$, 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, $n = 1$.

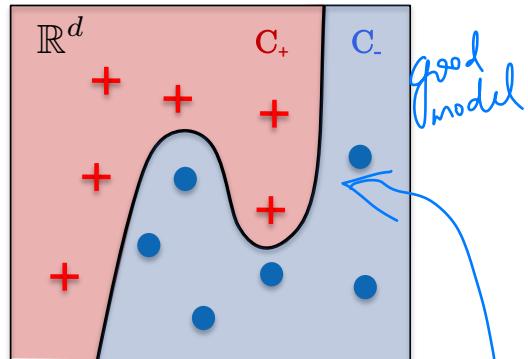
Geometric structure

• $\textcolor{red}{+}$ } Labeled

\blacktriangle Unlabeled

useful to infer the
data manifold

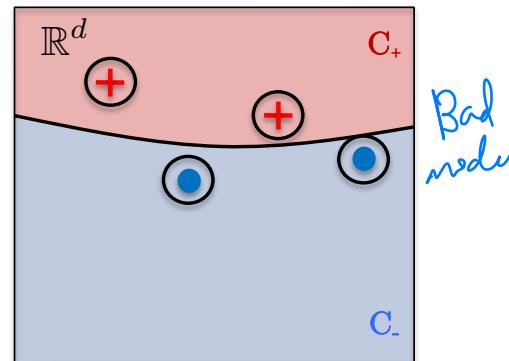
- Unlabeled data **encapsulate valuable statistical information**, particularly the **geometric structure** of the data distribution.
- How to leverage this information within the supervised SVM classification framework?



Labeled data $+$, ●
Supervised classification

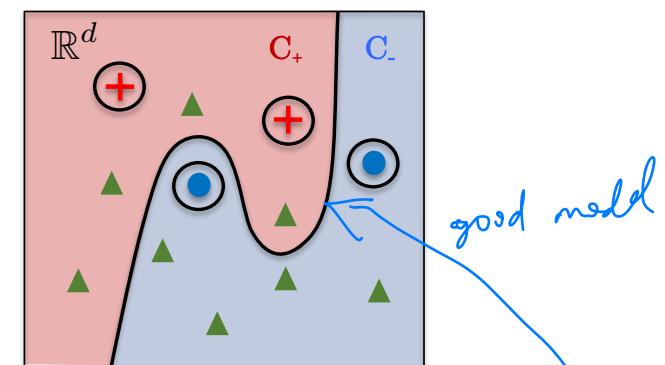
Ideal: Abundance of
unlabeled data

Kernel SVM solution
is good



Labeled data $+$, ●
Supervised classification

Lack of labeled data
Kernel SVM solution is
far from gt



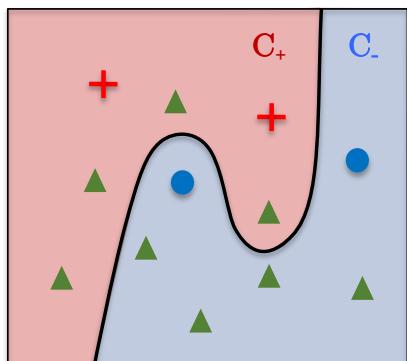
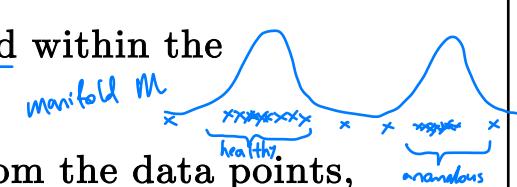
Labeled data $+$, ●
Unlabeled data \blacktriangle
Semi-supervised classification

Goal: Use the remaining
labeled data $(+)$ & (\bullet) and
the unlabeled data \blacktriangle to
build an accurate classifier
as if all datapoints were labeled

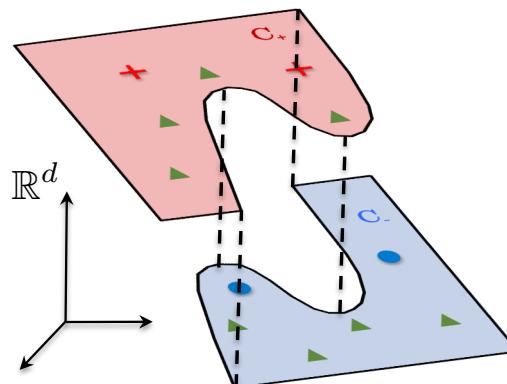
Manifold and graph

~~Key idea: Distribution of X remains unchanged whether or not we have Y .~~

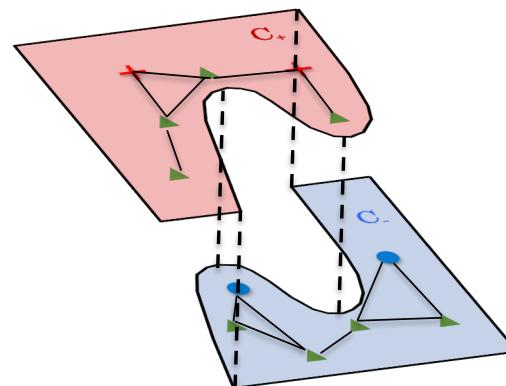
- The data distribution remains unchanged regardless of whether labels are present or absent.
- Both labeled and unlabeled data points are assumed to belong to the same manifold within the d -dimensional feature space.
- This manifold is estimated using a k -nearest neighbor graph constructed from the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



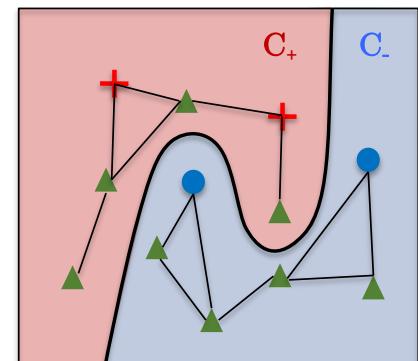
Labeled data $\textcolor{red}{+}$, $\textcolor{blue}{\circ}$
Unlabeled data \blacktriangle
Semi-supervised classification
on manifold



Manifold M embedded
in \mathbb{R}^d . Data points are
sampled from M .



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



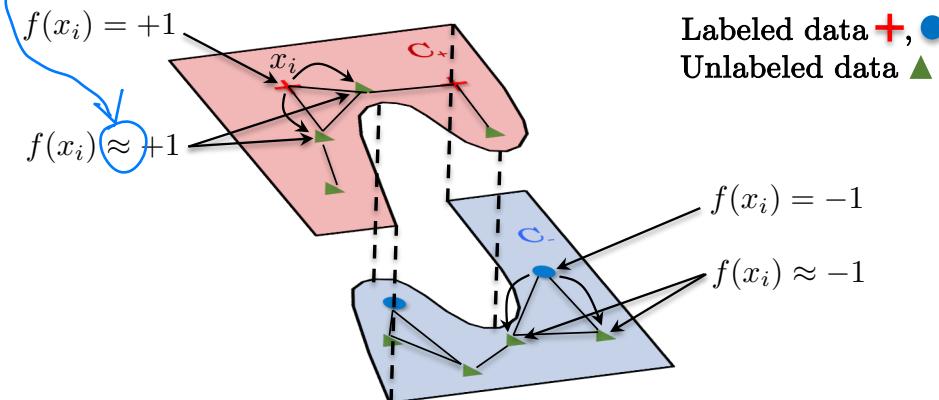
Graph $G = (V, E, A)$
 k -NN graph

smoothness
captured by
 ∇f

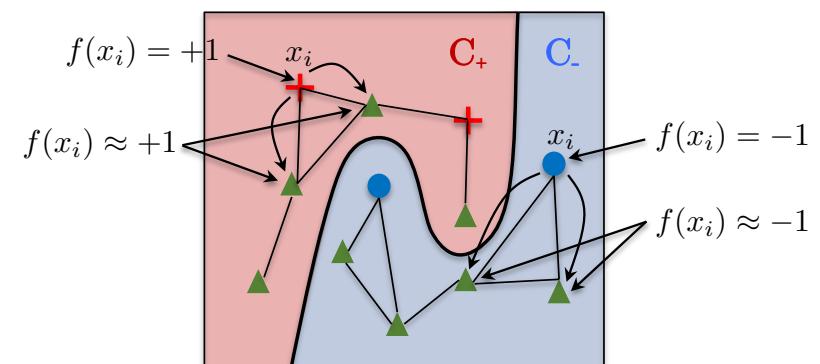
Manifold regularization

- 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.

! Laplacian is used for propagation



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



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

Graph regularization

Stochastic GD is a regularization technique?

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

"Roughness" of f
 \propto
("Smoothness" of f)^{-1}

Dirichlet energy of f =

$\Rightarrow \int_{\mathcal{M}} |\nabla f|^2$ *will be minimized* (continuous Dirichlet energy)

$\approx \sum_{ij \in V} A_{ij} |f(x_i) - f(x_j)|^2$ (discrete Dirichlet energy)

$\approx f^T \underbrace{\mathcal{L} f}_{\substack{\text{Normalized} \\ \text{Lsym}}} \in \mathbb{R}, \quad f \in \mathbb{R}^n, \quad \mathcal{L} = I - D^{-1/2} A D^{-1/2} \in \mathbb{R}^{n \times n}$ (Laplacian matrix)

$D = \text{diag}(d) \in \mathbb{R}^{n \times n}, d = A \mathbf{1}_n \in \mathbb{R}^n$ (degree vector)

- 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.

Wow $\min_f \int_{\mathcal{M}} |\nabla f|^2 \iff \text{making } f \text{ smooth} \iff [A_{ij}=1 \rightarrow f(x_i) \approx f(x_j)]$

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

Semi-supervised classification with graphs

- SSC optimization problem with graph smoothness :

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

This is the SVM objective

classification error

Regularization smoothness term
 proxy measurement for the roughness in the signal f .

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

with

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



Misha Belkin

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

Optimization algorithm

- Dual optimization problem :

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

with

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

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

with $Q = LHK(I + \gamma \mathcal{L}K)^{-1}HL \in \mathbb{R}^{n \times n}$

Solution : $\xi^* = (I + \gamma \mathcal{L}K)^{-1}HL\alpha^*$

T in Lab 4 code L: label

H: ?

K: Kernel matrix

L: Normalized Laplacian
 $L_{\text{sym}} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$

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

Run Graph SVM

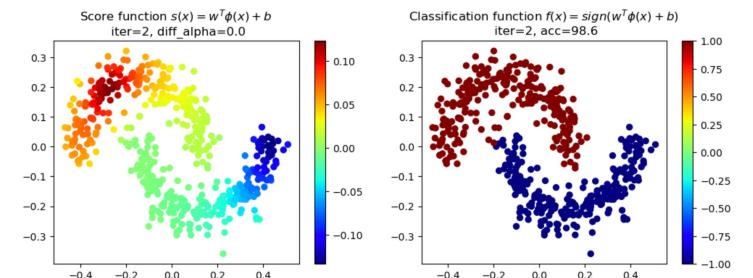
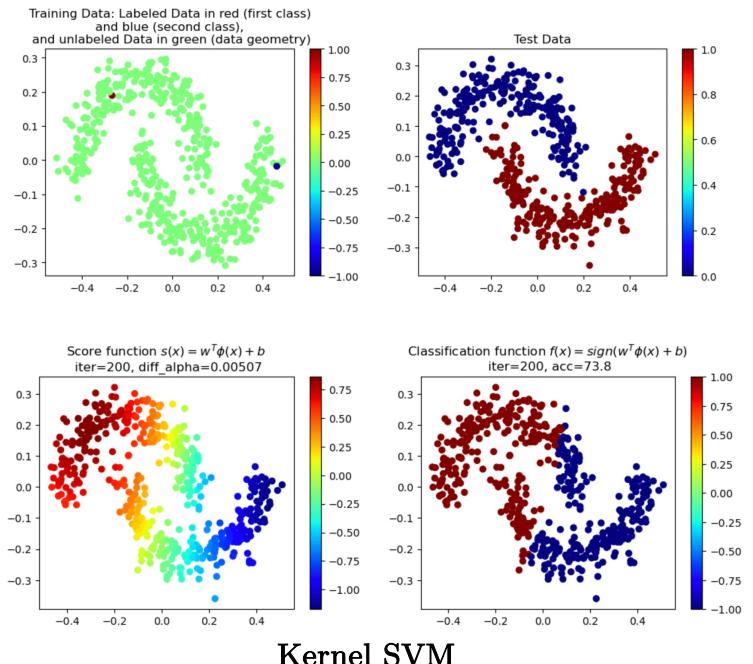
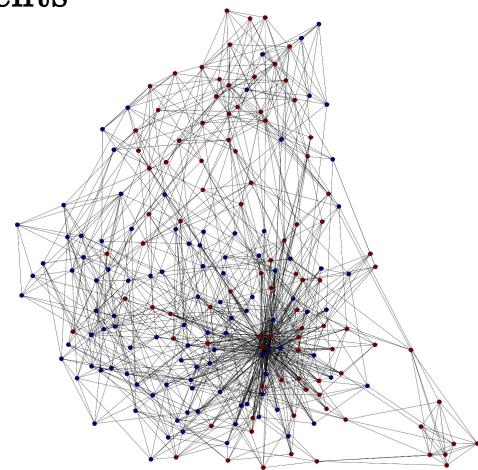
```
# Run Graph SVM

# 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)
Ktest = 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
```



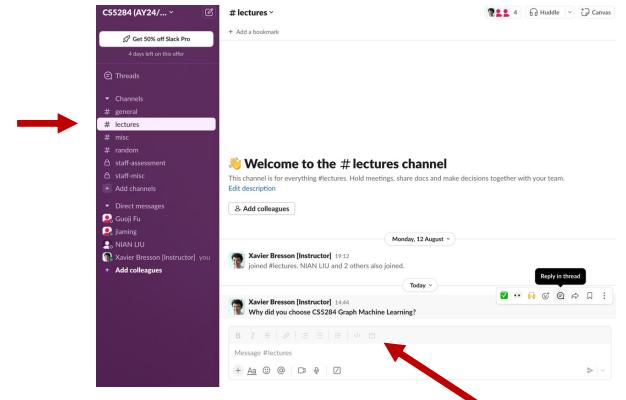
Graph SVM

In-lecture question

- Propose an alternative graph regularization loss to the Dirichlet loss defined as:

$$\int_{\mathcal{M}} |\nabla f|^2 \approx f^T L f = \sum_{ij \in V} A_{ij} |f(x_i) - f(x_j)|^2$$

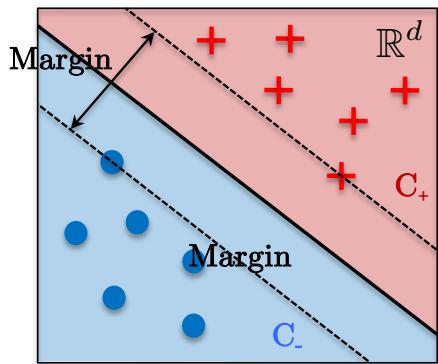
- In Slack #lectures
 - Identify the question and Reply in thread with a short response
- Answer :



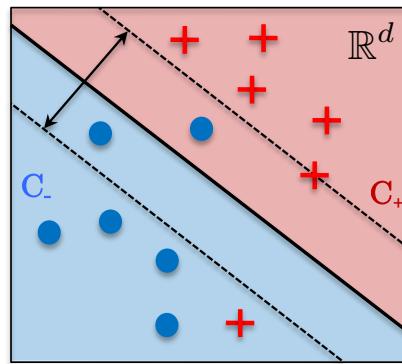
Outline

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

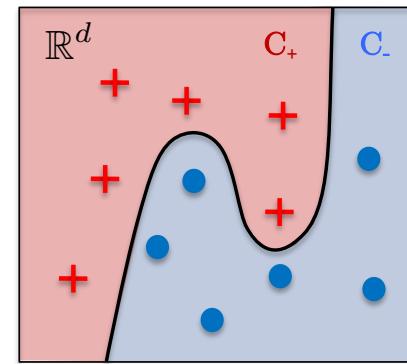
History of SVM techniques



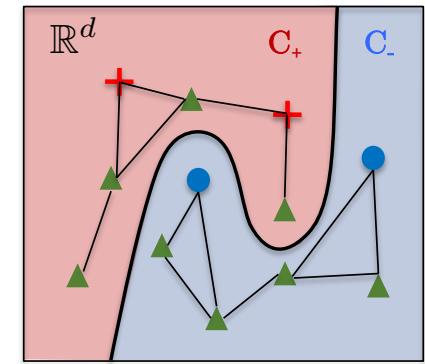
Linear SVM^[1]
Supervised learning



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 :

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

where

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

What's the diff ?

with

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

All can be used as regularizers to the optimization prob.

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



Questions?