



3D geoinformation

Department of Urbanism  
Faculty of Architecture and the Built Environment  
Delft University of Technology

GEO5017

Machine Learning for the Built Environment

Lecture

# Support Vector Machine

Shenglan Du




# Today's Agenda

- Previous Lecture: Classification
- Support Vector Machine
  - Standard SVM
  - Soft Margin SVM
- SVM Applications



# Today's Agenda

- Previous Lecture: Classification 
- Support Vector Machine
  - Standard SVM
  - Soft Margin SVM
- SVM Applications

# Classification



- We usually have a set of input data represented as feature vectors:

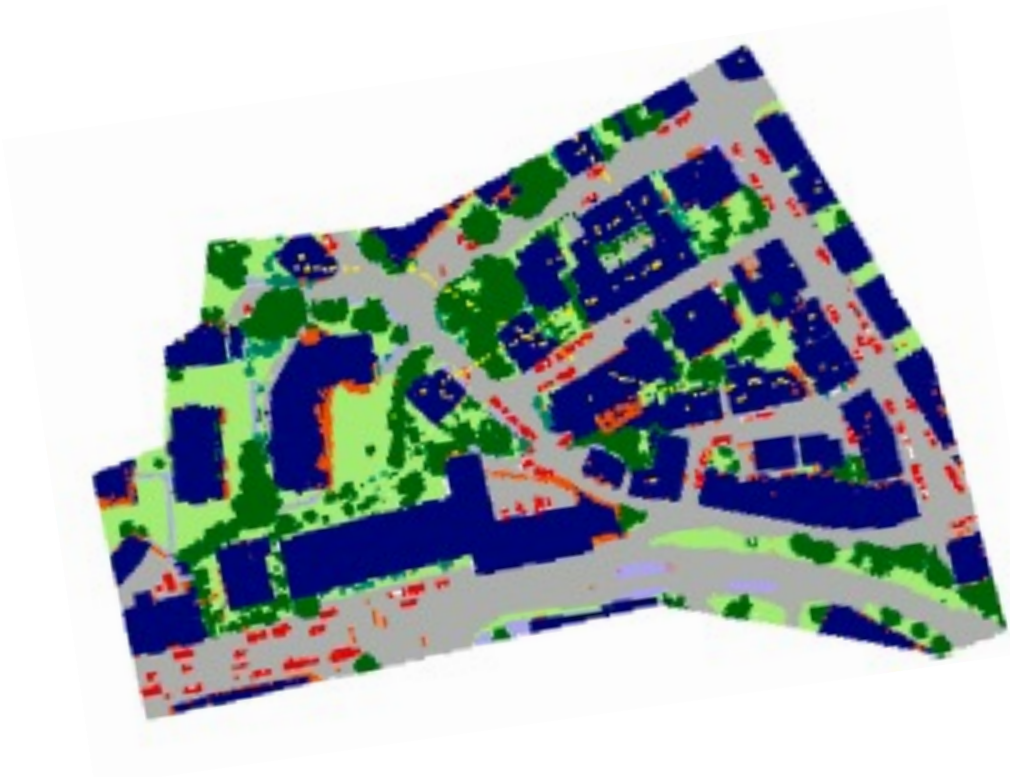
$$\mathbf{x} = (x_1, x_2, x_3 \dots x_p)^T$$

- Classification aims to specify which category/class  $\mathbf{y}$  some input data  $\mathbf{x}$  belong to





# Classification



- An example of point cloud semantic classification



$$\mathbf{x} = (x, y, z, r, g, b, intensity \dots)^T$$

- $\mathbf{y}$ :
-  High vegetation
  -  Low vegetation
  -  Building
  -  Road
  -  Grass land

# Classification



- Two classification approaches:
  - **Generative approach**: model the probability distribution of feature  $\mathbf{x}$  and label  $\mathbf{y}$ 
    - Bayes classifier
    - Gaussian mixture model
  - **Discriminative approach**: model a function that directly map from feature  $\mathbf{x}$  to label  $\mathbf{y}$ 
    - Linear classifier (Fisher, Logistic regression, SVM)
    - Non-linear classifier (Decision tree, Neural networks)

# Classification



- Many classification or regression problems can be specified as:
  - Find a suitable model / hypothesis
  - Define a loss function (i.e., least squares, maximum likelihood ...)
  - Feed the data samples into the model and search for the model parameters that cause the least loss

# Classification



- Standard linear (Fisher) classifier:
  - **Hypothesis**: the decision boundary is a linear model of the input vector  $\mathbf{x}$ :

$$\mathbf{w}^T \mathbf{x} + b = 0$$

- **Loss Function**: least squares
- Logistic regression:
  - **Hypothesis**: the posterior probability is a logistic sigmoid of a linear function of  $\mathbf{x}$


$$P(y|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

- **Loss Function**: maximum likelihood





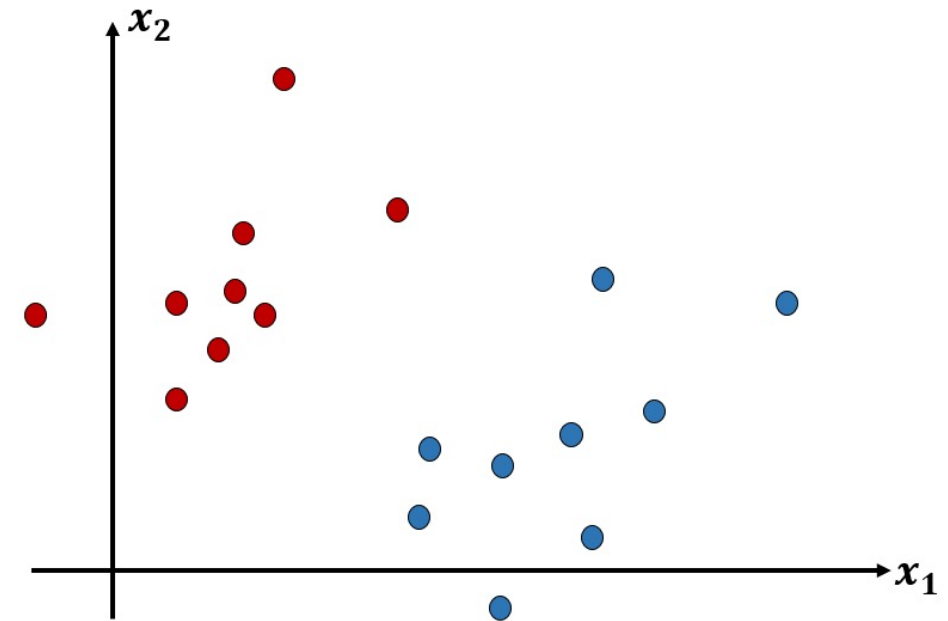
# Today's Agenda

- Previous Lecture: Classification
- **Support Vector Machine** 
  - Standard SVM
  - Soft Margin SVM
- SVM Applications

# Support Vector Machine

- We consider two-class (+1, -1) linearly separable task
- Constrain the weights so that the output is always larger than 1 or smaller than -1

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq +1 & \text{if } y_i = +1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 & \text{if } y_i = -1 \end{cases}$$



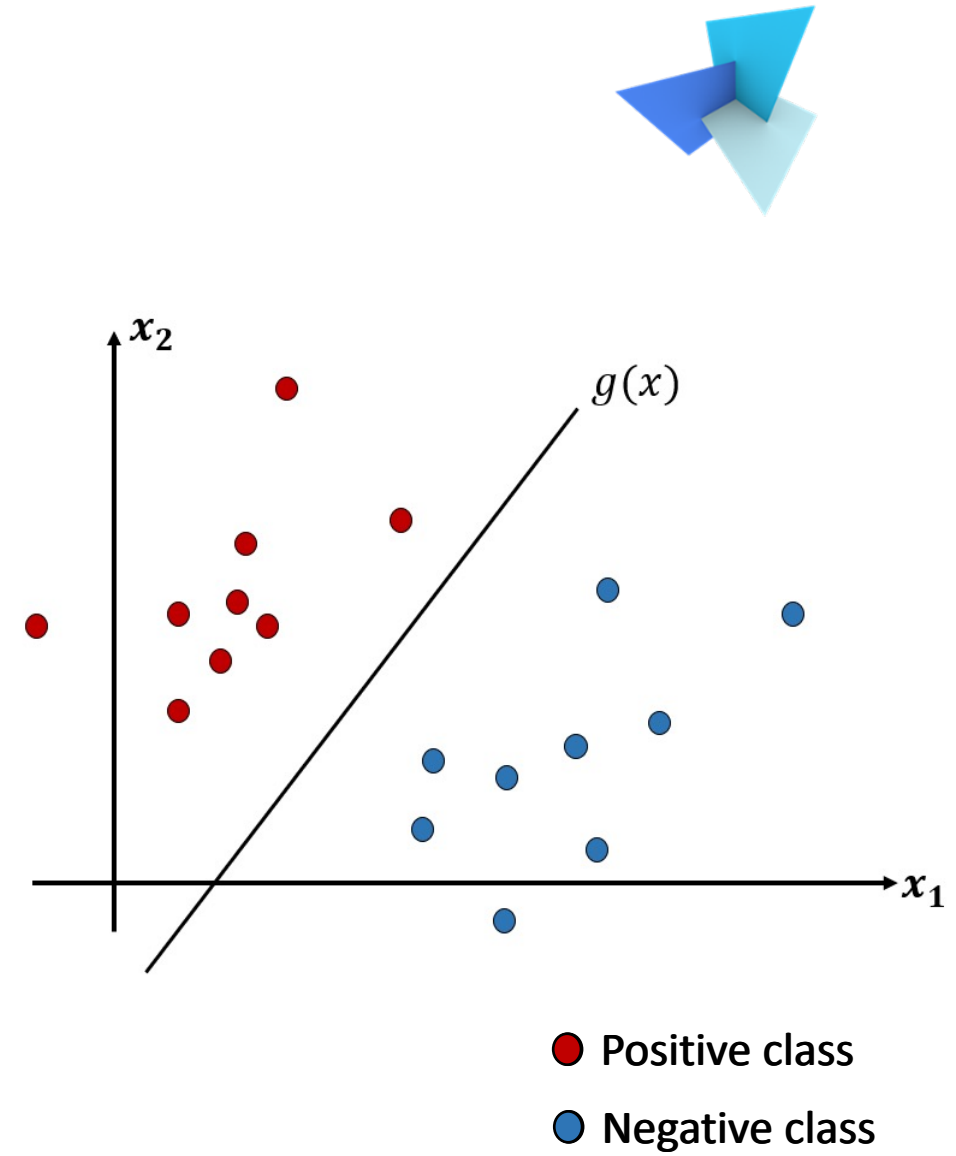
- Positive class
- Negative class



# Support Vector Machine

- Given a set of data samples, we aim to find a decision boundary for the input vector space:

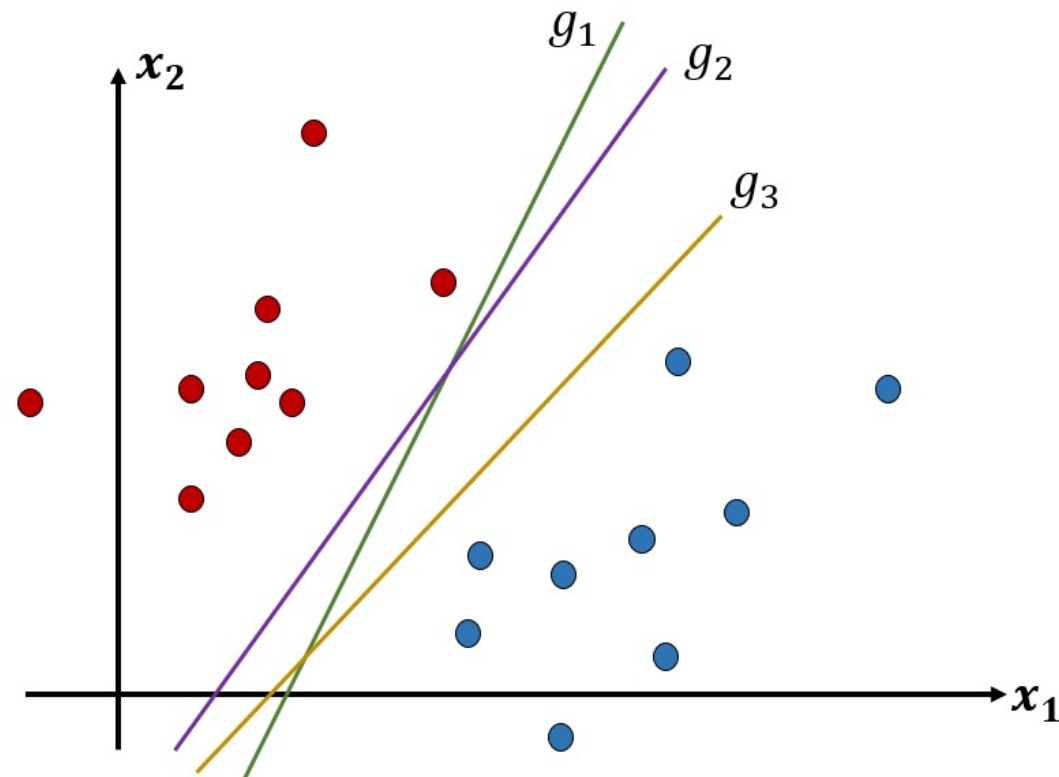
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = 0$$



# Support Vector Machine



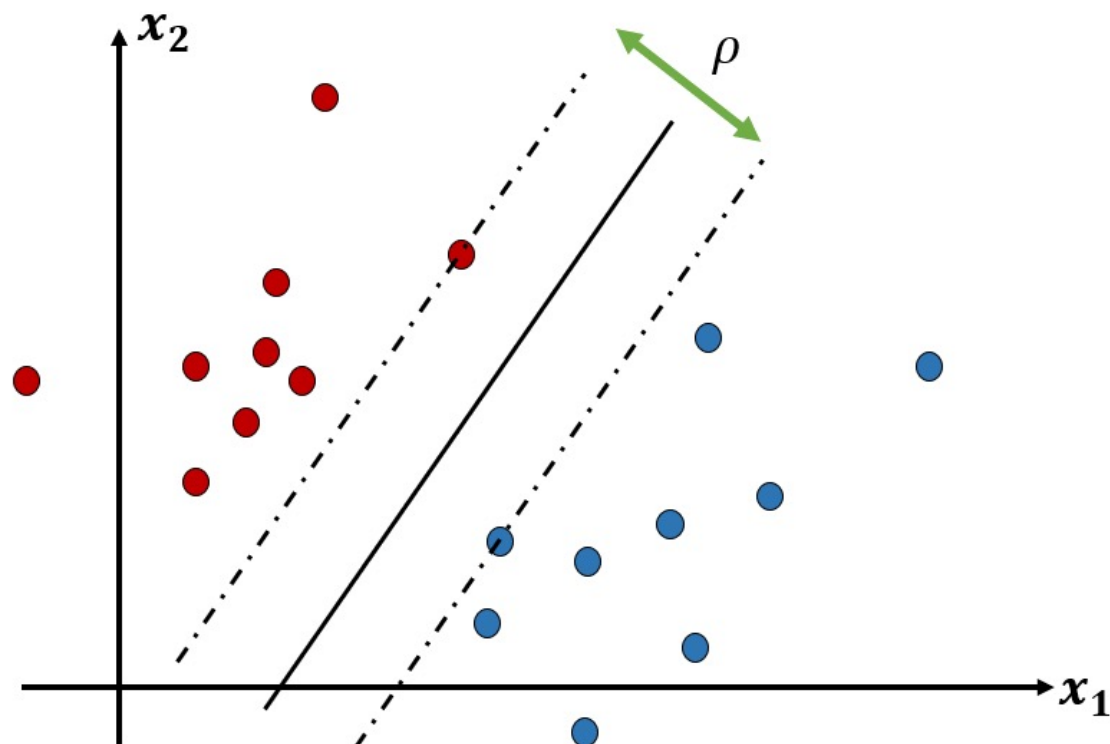
- Such a decision boundary is not unique






# Support Vector Machine

- The goal of SVM is to find a decision boundary that gives the maximum possible margin  $\rho$



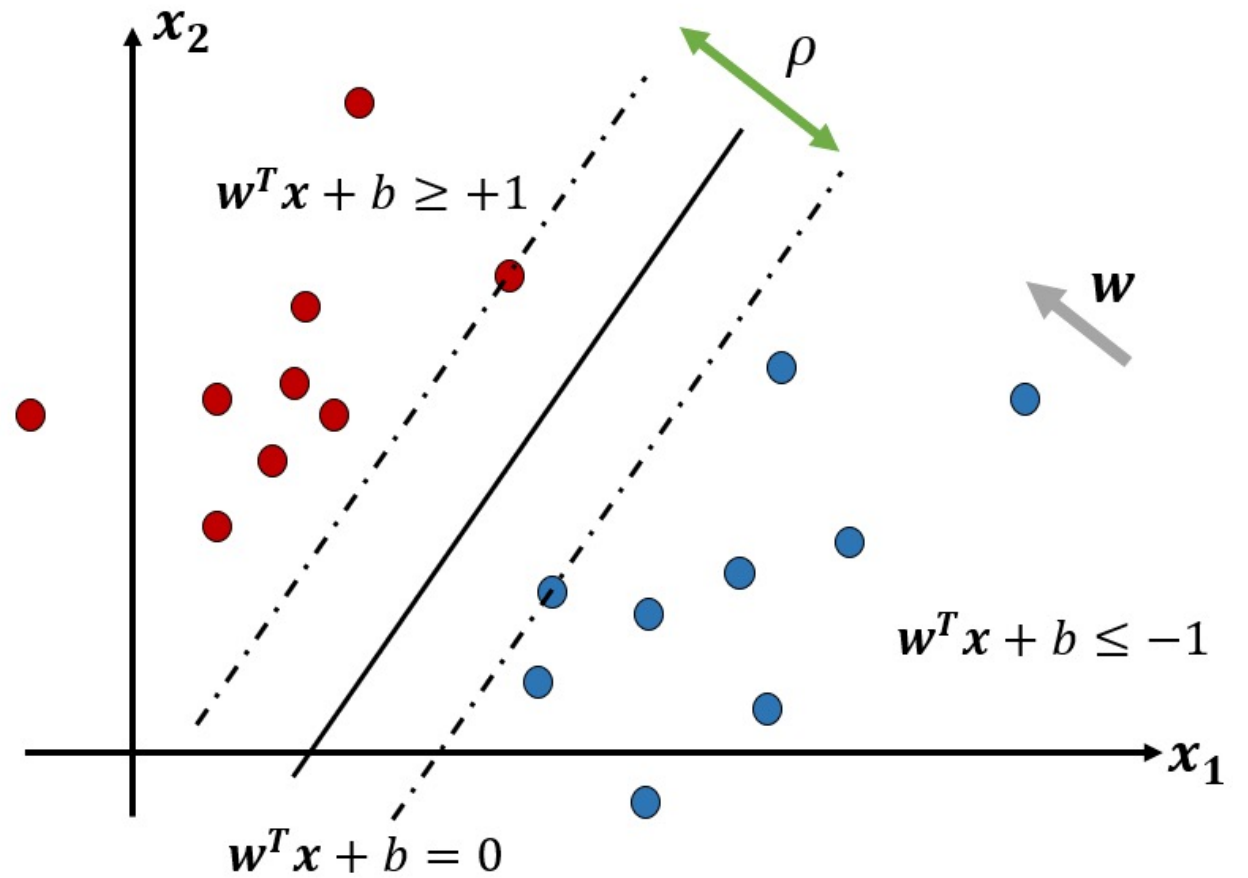


# Today's Agenda

- Previous Lecture: Classification
- Support Vector Machine
  - Standard SVM 
  - Soft Margin SVM
- SVM Applications

# Standard SVM

- What is the margin  $\rho$  ?



# Standard SVM

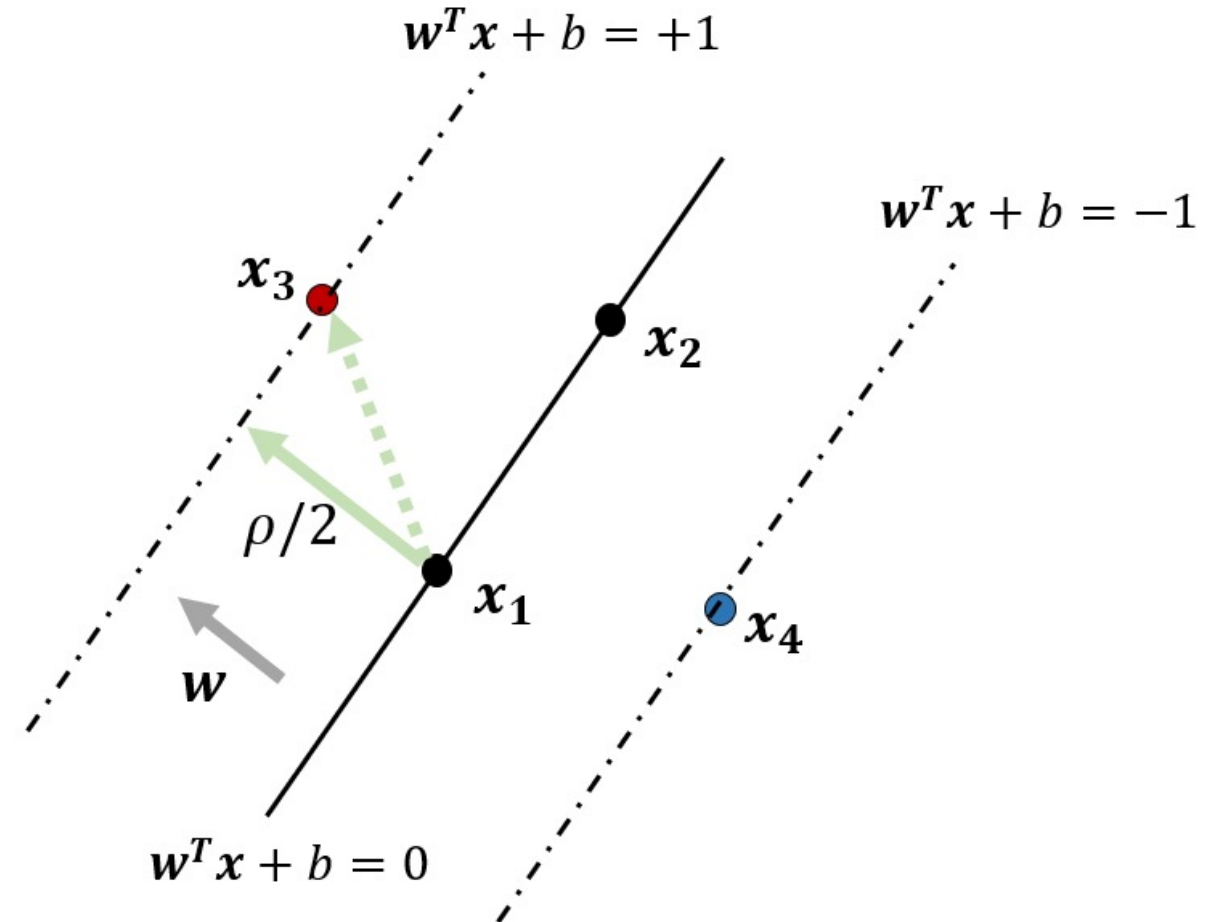


- $w$  is orthogonal to the line

$$w^T(x_2 - x_1) = 0$$

- $\rho/2$  is the projection of  $(x_1, x_3)$  over  $w$

$$\rho/2 = \frac{w^T(x_3 - x_1)}{\|w\|} = \frac{1}{\|w\|}$$



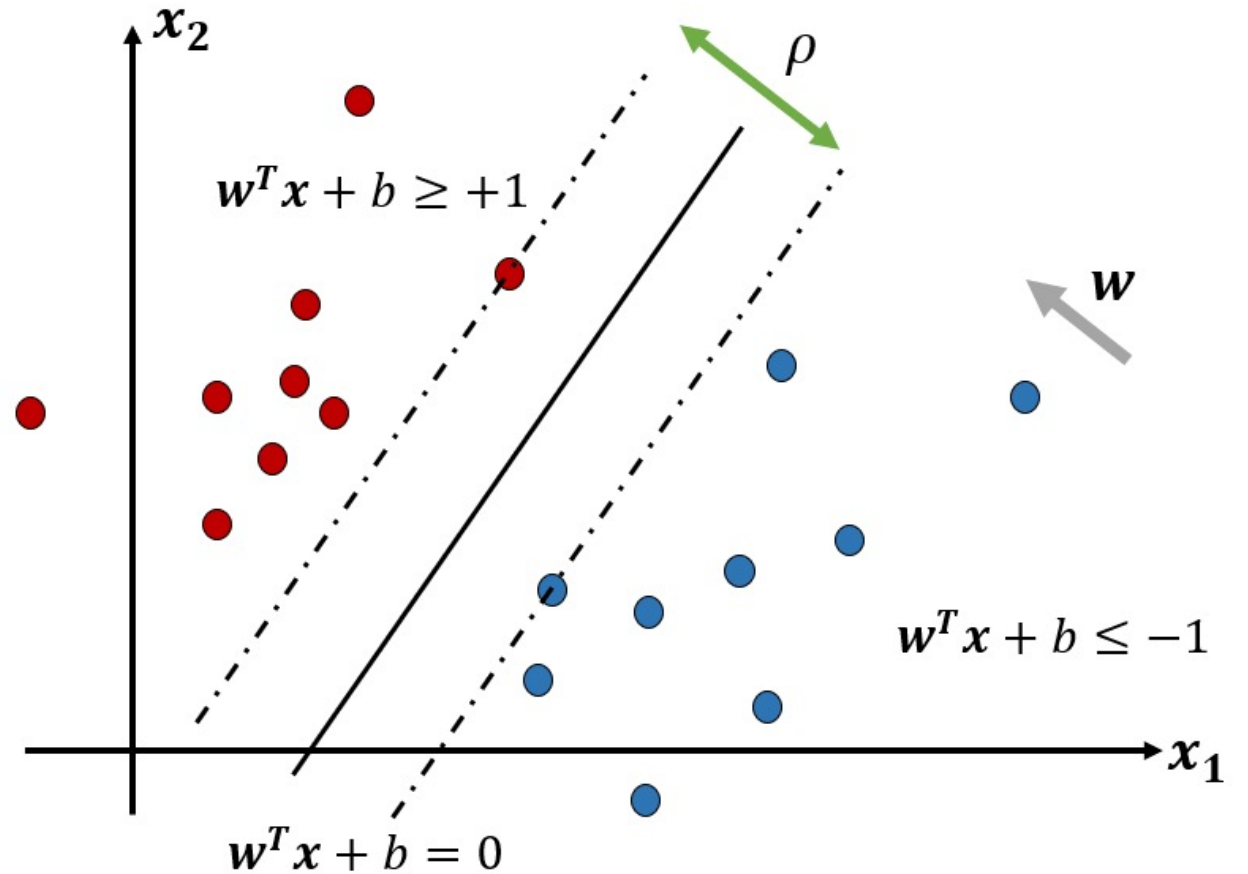


# Standard SVM



- The margin  $\rho$  can be represented as:

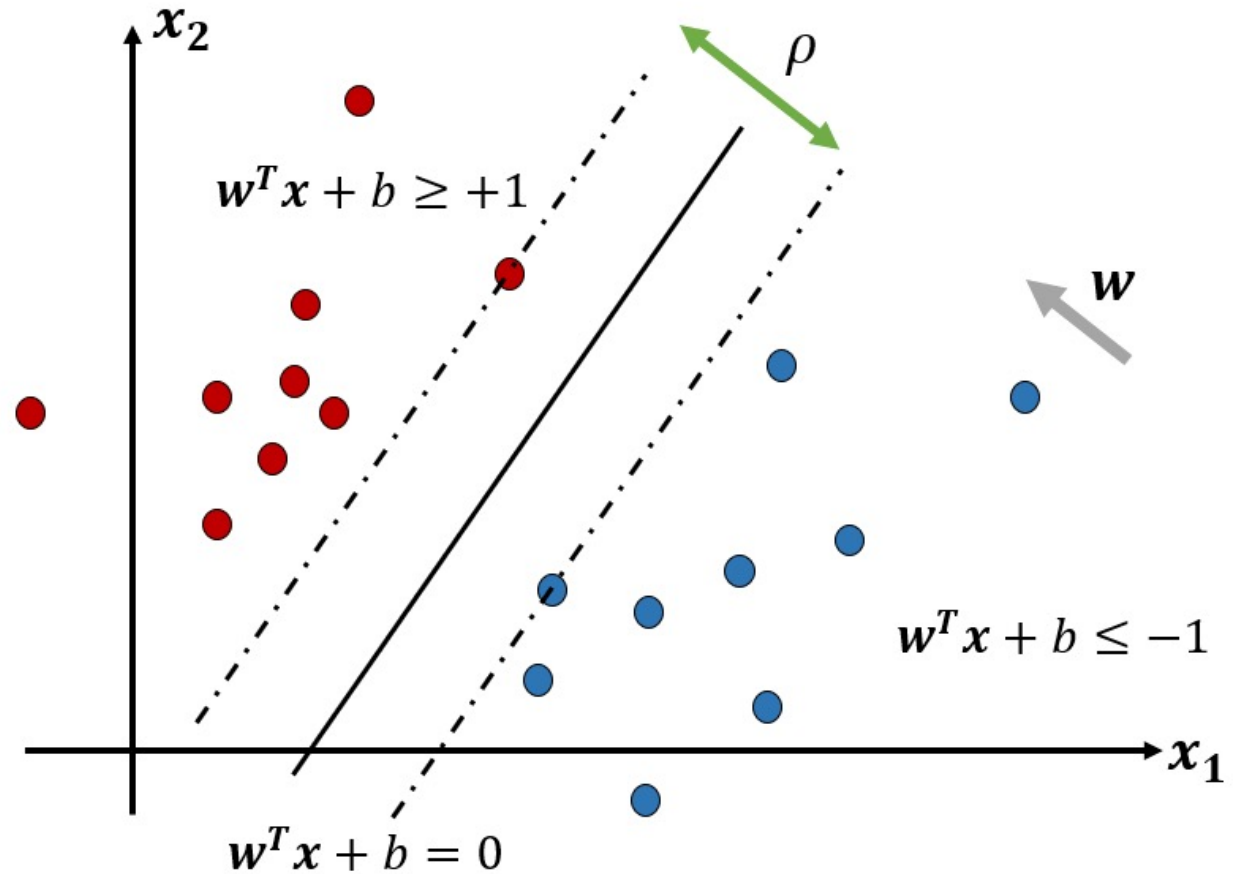
$$\rho = \frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|}$$



# Standard SVM

- SVM aims to solve:

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$
$$s.t. \begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq +1 & \text{if } y_i = +1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 & \text{if } y_i = -1 \end{cases}$$



# Standard SVM

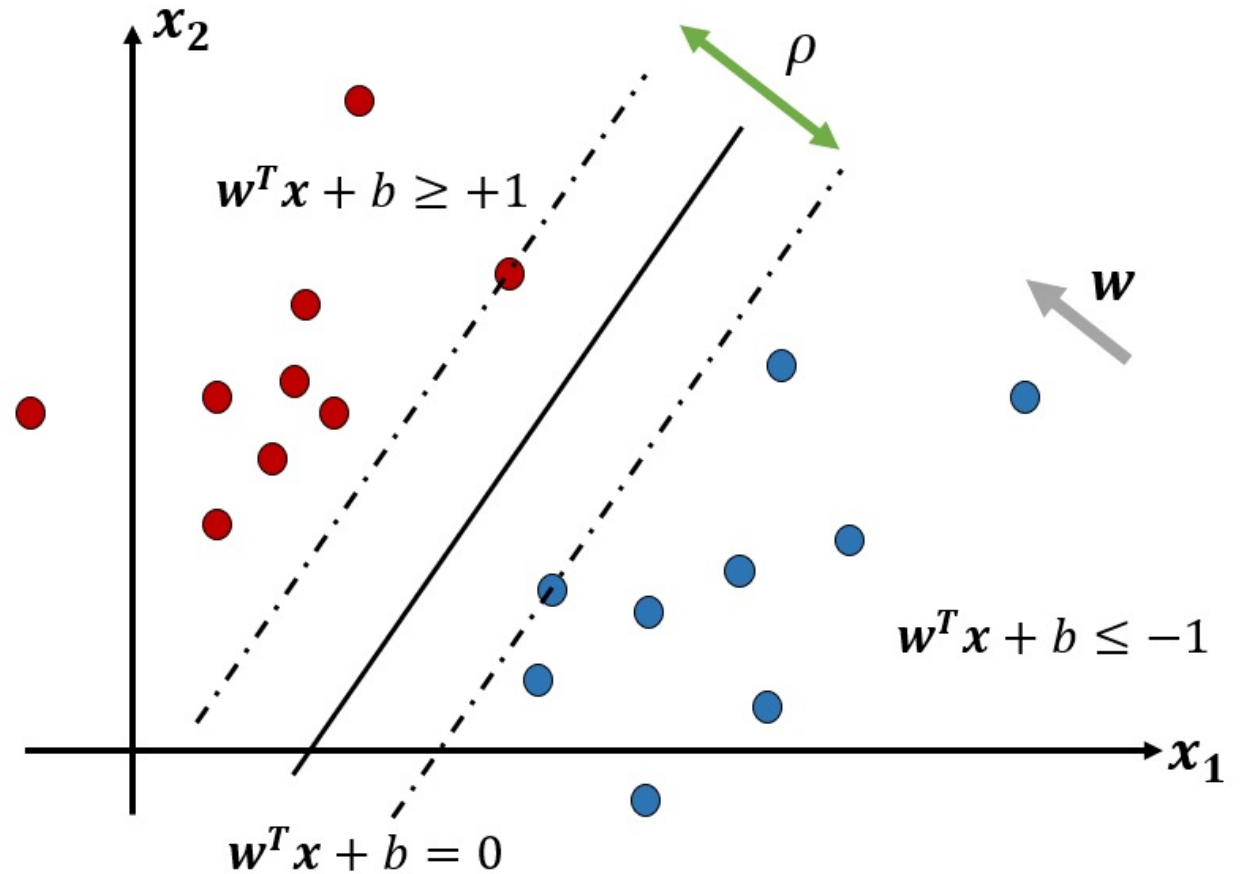
- SVM aims to solve:

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$

$$s.t. \begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq +1 & \text{if } y_i = +1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 & \text{if } y_i = -1 \end{cases}$$

Can be rewritten as:

$$s.t. y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0$$



# Standard SVM: Optimization



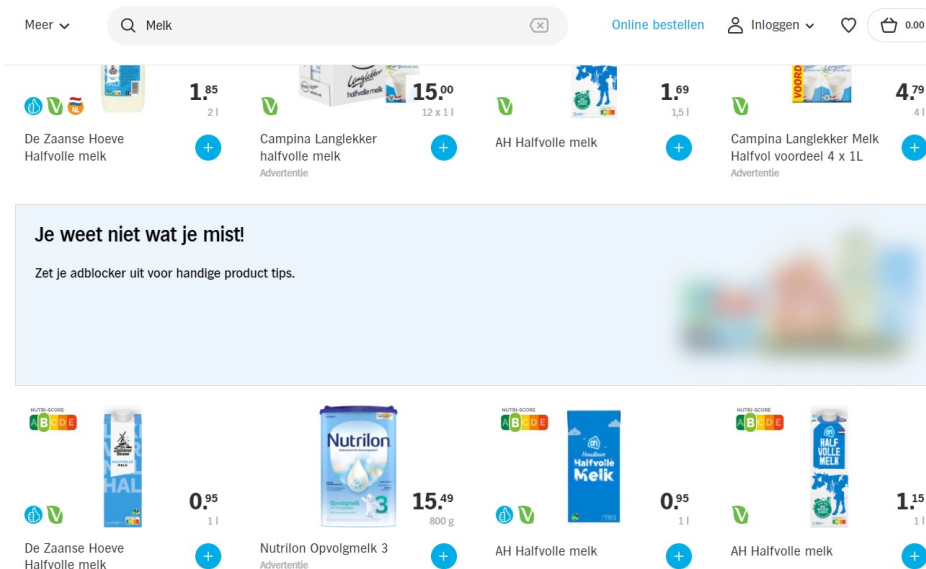
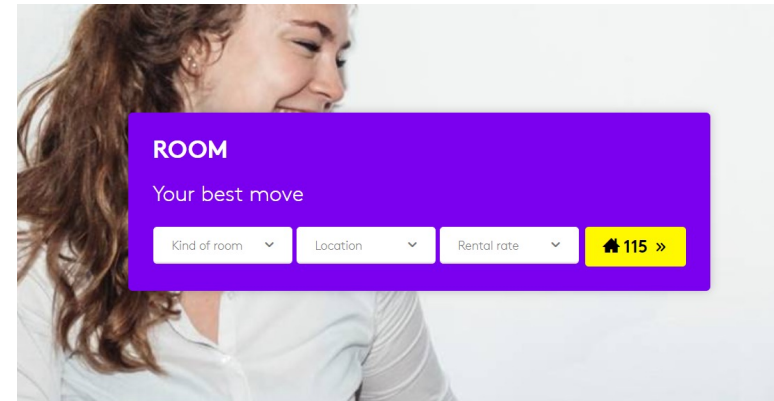
- What is optimization



# Standard SVM: Optimization



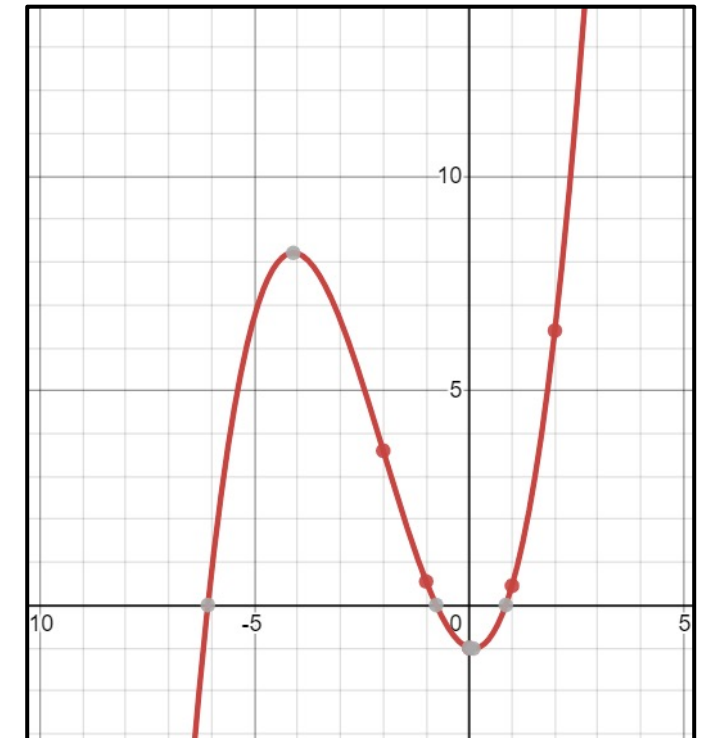
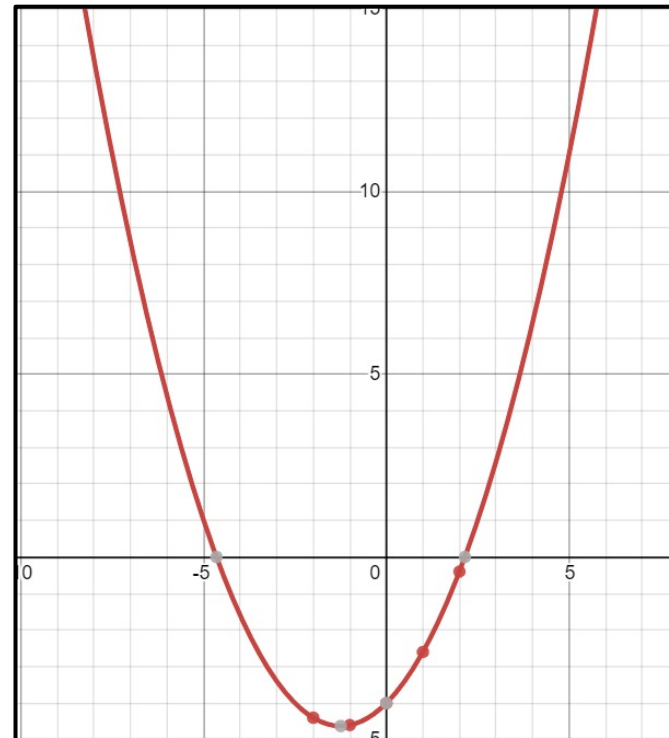
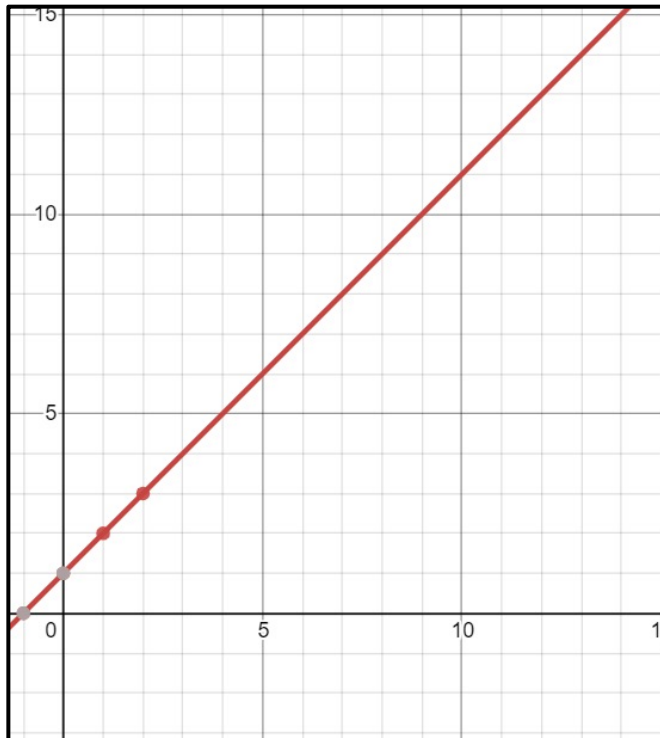
- Discrete options
  - Choose a brand of milk
  - Choose where to rent
  - Choose to study or to browse facebook this afternoon



# Standard SVM: Optimization



- Continuous space





# Standard SVM: Optimization

- What is optimization?
  - A goal (i.e., maximization, minimization)
  - Some constraints
  - Sometimes constraints might be compromised to achieve the goal
- SVM optimization problem:

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0, \forall i = 1, 2, 3, \dots, n$$



# Standard SVM: Optimization

- A constrained optimization problem can often be solved with Lagrangian (*this will not be asked in the exam*)

$$L(\mathbf{w}, \lambda) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \lambda_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1)$$





# Standard SVM: Optimization

- Computing the derivative and making it to 0 we get:

$$\frac{\partial L(\mathbf{w}, \lambda)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i = \mathbf{0}$$
$$\mathbf{w} = \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i$$

- Solving  $\lambda_i$  requires Quadratic programming and thus is not covered in this lecture



# Standard SVM: Optimization

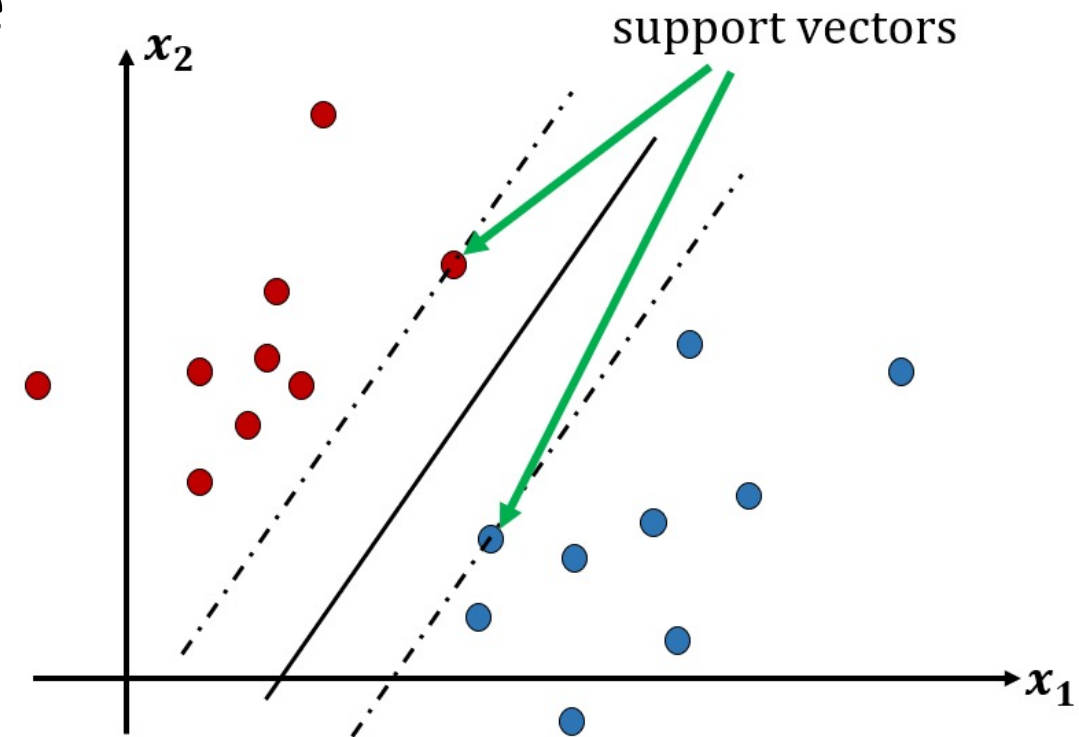
$$\mathbf{w} = \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i$$

- After solving the problem, a lot of  $\lambda_i$  becomes 0
- Only the objects  $\mathbf{x}_i$  with non-zero  $\lambda_i$  contribute to  $\mathbf{w}$
- These objects are called the “**support vector**”

# Standard SVM: Optimization



- Support vectors usually lie near the boundary
- Objects far away from boundary will have little influence on the model





# Standard SVM

- Overview:
  - **Hypothesis**: the decision boundary is a linear model of the input vector  $\mathbf{x}$ :

$$\mathbf{w}^T \mathbf{x} + b = 0$$

- **Loss Function**:


$$\min \frac{1}{2} \|\mathbf{w}\|^2$$

$$s.t. \ y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0$$

- **Aim**: find a decision boundary that gives the maximum margin



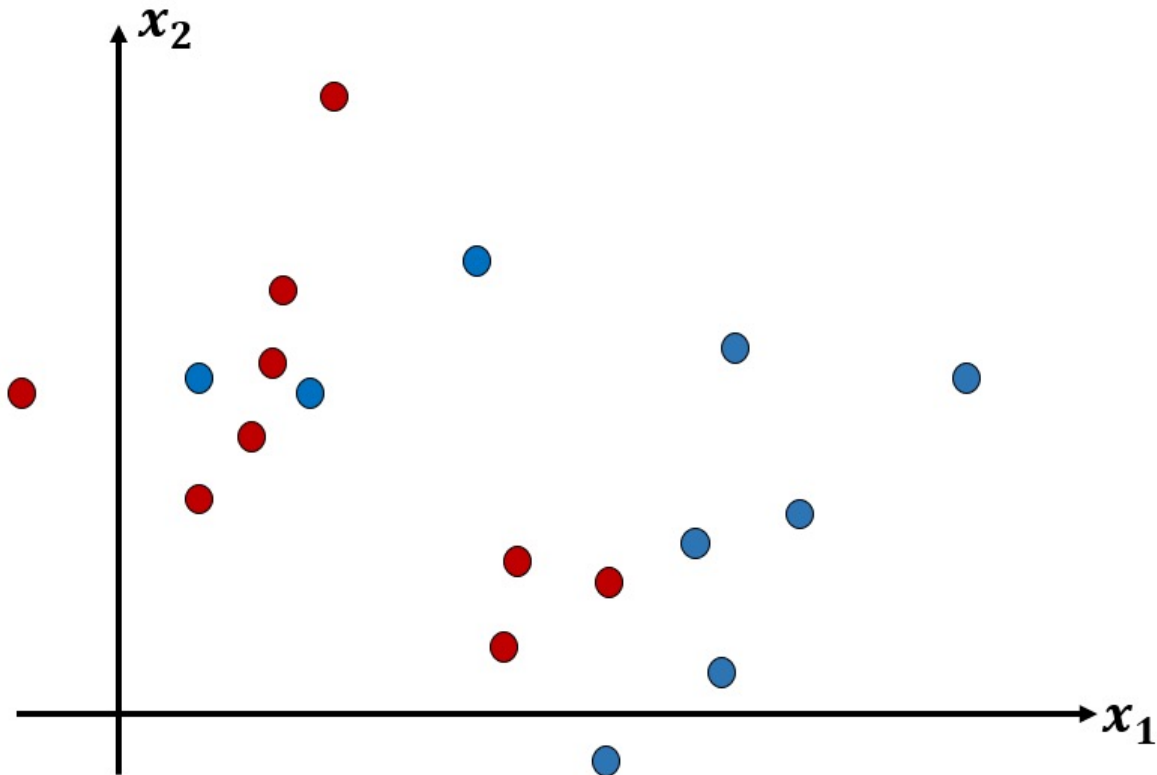
# Today's Agenda

- Previous Lecture: Classification
- Support Vector Machine
  - Standard SVM
  - Soft Margin SVM 
- SVM Applications

# Soft Margin SVM



- When classes are not linearly separable .....

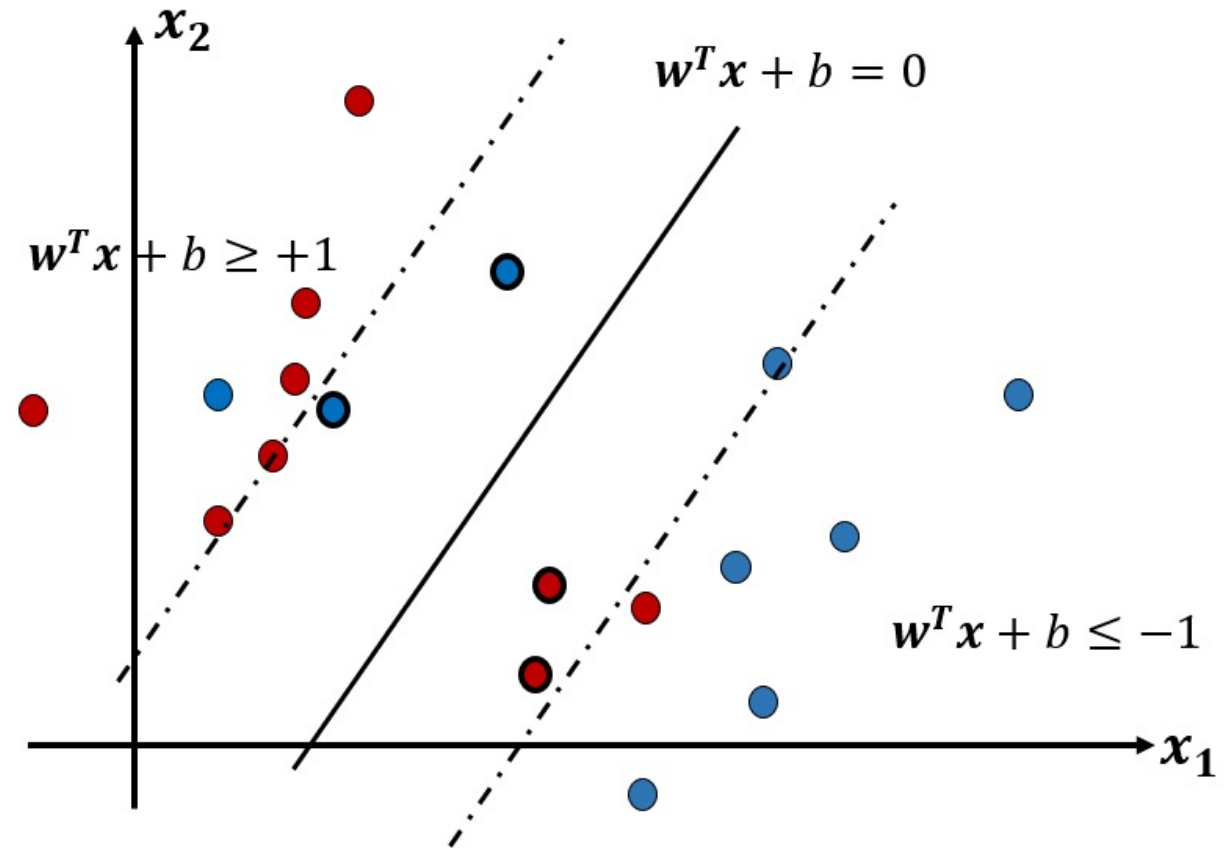


# Soft Margin SVM



- Standard SVM will cause misclassifications
- For misclassified samples:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \leq 1$$





# Soft Margin SVM

- We introduce a new slack variable  $\xi_i$ ,  $i=1,2,\dots,n$ , which refers to the soft margin
- Soft margin SVM aims to solve:

$$\begin{aligned} \min & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} & y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

Where C is a constant term




# SVM Remarks



- Strengths:
  - It generalizes well in high-dimensional space with relatively low sample sizes
  - It is little affected by data distribution / density
- Weaknesses:
  - It is usually computational expensive
  - It performs bad when classes are highly overlapped (although the soft-margin tricks have been developed to cope with overlapping issue)



# Today's Agenda

- Previous Lecture: Classification
- Support Vector Machine
  - Standard SVM
  - Soft Margin SVM
- SVM Applications 

# SVM Applications



- [1] Eigen-feature analysis of weighted covariance matrices for LiDAR point cloud classification (Lin et al, 2014, ISPRS Journal)

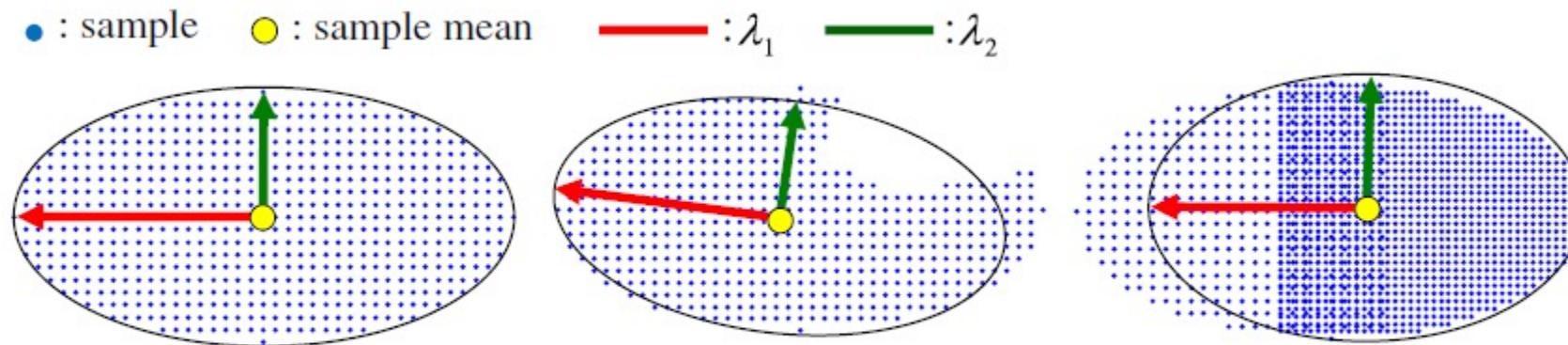
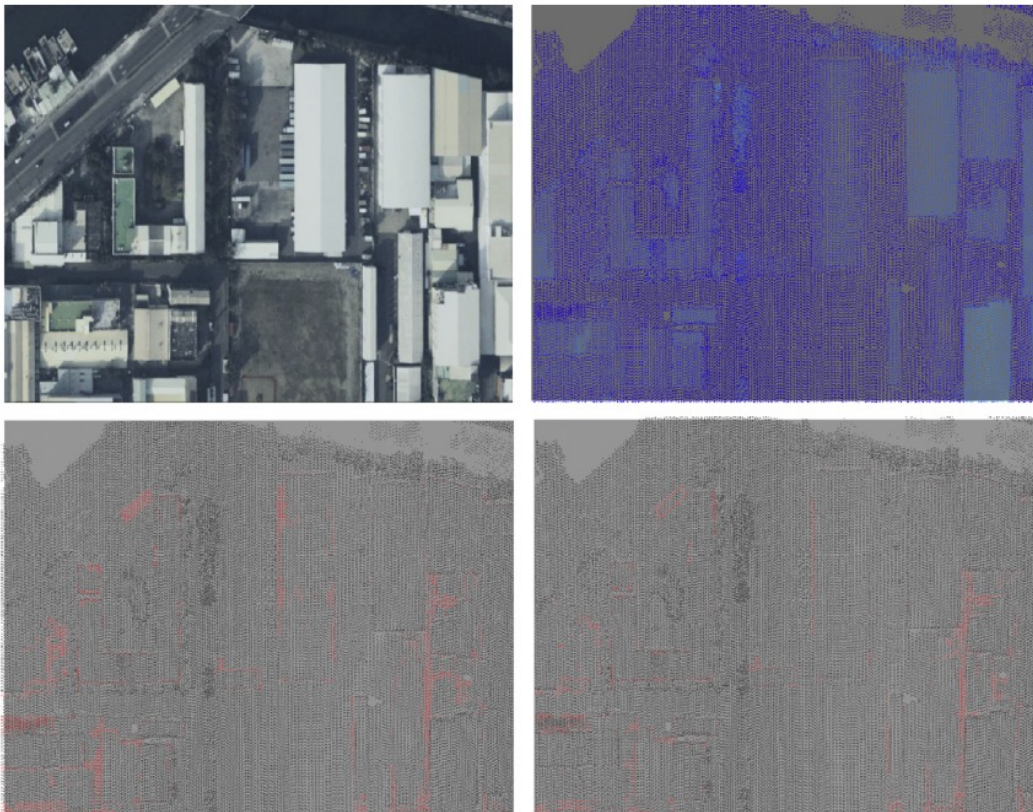


Fig. 1. Results of PCA on point datasets with uniform sampling (top), incomplete surface sampling (middle), and different sampling densities (bottom).

# SVM Applications



- [1] Eigen-feature analysis of weighted covariance matrices for LiDAR point cloud classification (Lin et al, 2014, ISPRS Journal)

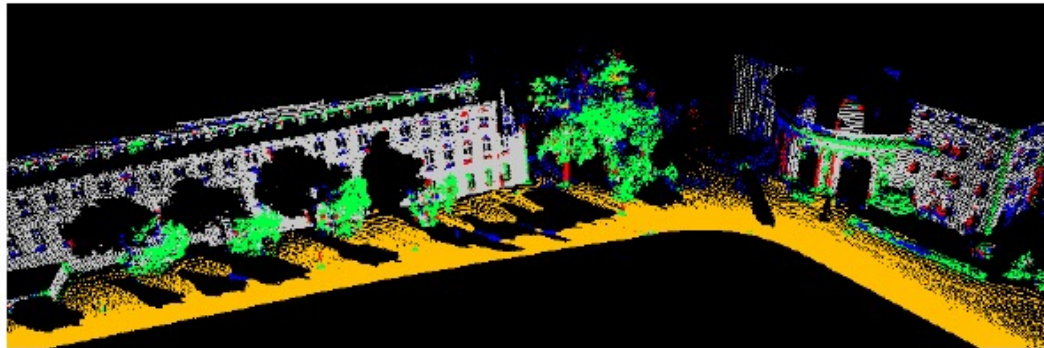


Top left: orthoimage; top right: point clouds;  
Bot left: classification results from standard eigen features  
Bot right: classification results from generated features

# SVM Applications

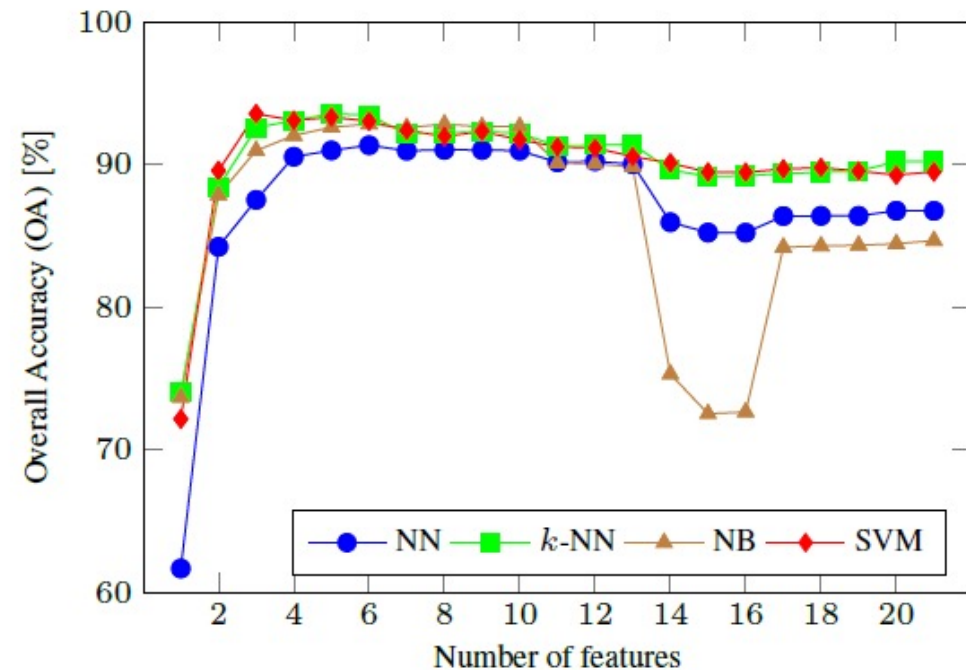


- [2] Feature relevance assessment for the semantic interpretation of 3D point cloud data. (Weinmann et al, 2013, ISPRS Annals)



	NN	$k$ -NN	NB	SVM
all features	86.74	90.21	84.66	89.48
only 3D features	85.70	88.64	85.80	88.34
only 2D features	82.66	83.80	77.09	80.06
eigenvalue-based 3D features	80.36	82.42	83.92	84.81
$\{L_\lambda, P_\lambda, S_\lambda\}$	62.91	75.72	76.90	75.26
five best-ranked features	90.97	93.53	92.61	93.32

Table 3: Overall classification accuracies in % for different feature subsets and different classifiers.







# SVM Applications

- Steps to apply SVM and other classical classifiers:
  - Collect raw data
  - Design and compute features
  - Feed the data and features to train the model
  - Test and evaluate the model
- Most of the traditional machine learning works focus on feature engineering



Questions?



3D geoinformation

Department of Urbanism  
Faculty of Architecture and the Built Environment  
Delft University of Technology

GEO5017

Machine Learning for the Built Environment

Lab Session

# SVM Practice in Scikit Learn

Shenglan Du



# Scikit-Learn



## scikit-learn

Machine Learning in Python

Getting Started

Release Highlights for 1.0

GitHub

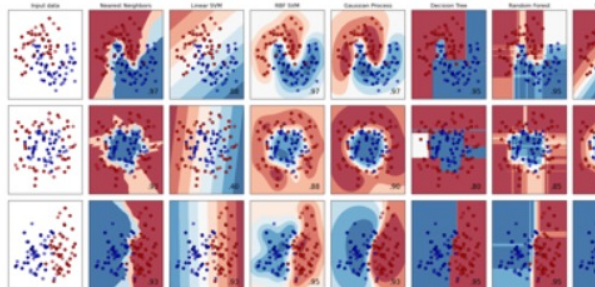
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

## Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, and more...

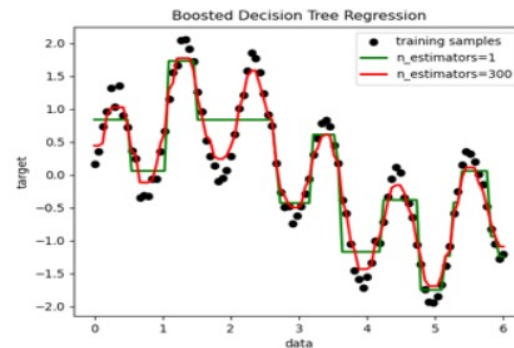


## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, nearest neighbors, random forest, and more...



## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, and more...

K-means clustering on the digits dataset (PCA-reduced data)  
Centroids are marked with white cross



# Scikit-Learn



- Installation with pip or conda

Installing the latest release

Operating System

Windows

macOS

Linux

Packager

pip

conda

Use pip virtualenv

Install the 64bit version of Python 3, for instance from <https://www.python.org>.

Then run:

```
$ pip install -U scikit-learn
```



# SK-Learn for SVM

- SVM has three classifiers in the library
  - **SVC**: most commonly used in practice
  - **NuSVC**: similar to SVC, has slightly different yet equivalent mathematical formulations and parameter set
  - **LinearSVC**: faster implementation of SVM, but can only adopt linear kernels

# SVC: Getting started



```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
SVC()
```

After being fitted, the model can then be used to predict new values:

```
>>> clf.predict([[2., 2.]])
array([1])
```

- X is input samples with features, y is the labels
- Model.fit() gives the trained model
- Model.predict() gives the predictions on other unseen samples



# SVC: Documentation

## sklearn.svm.SVC

```
class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None)
```

[\[source\]](#)

- The most important hyper-parameters:
  - C: the coefficient introduced in soft-margin SVM
  - Kernel: a trick you can use to transform input features



# SVC: Documentation

## sklearn.svm.SVC

```
class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None)
```

[\[source\]](#)

- Other important hyper-parameters:
  - `Class_weight`: specify the weight per class. You either input a dictionary of pre-fixed weights, or use 'balanced'.
  - `Max_iter`: hard limit on iterations within solver, or -1 for no limit.
  - `Decision_function_shape`:
    - 'ovr': default, one versus the rest for multi-class
    - 'ovo': one versus one for multi-class



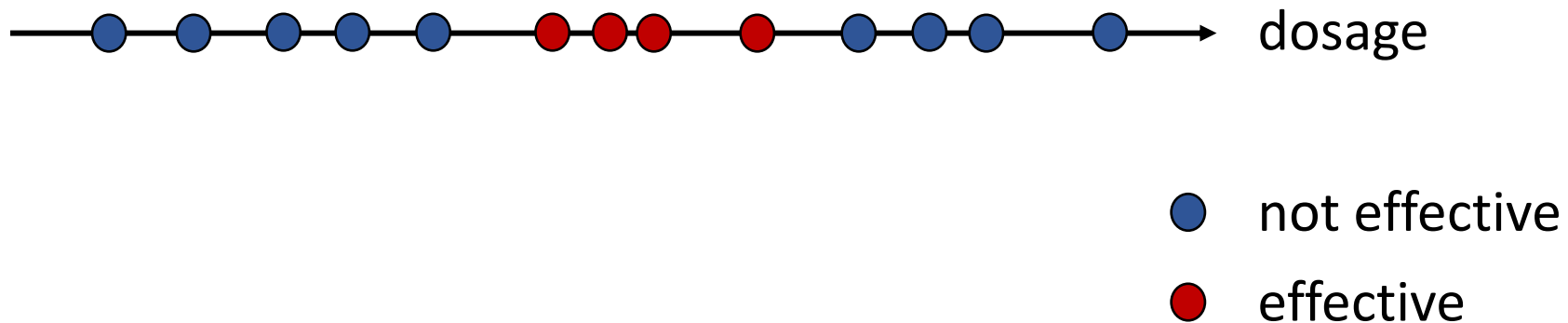
# SVC: Interesting Attributes

- `n_features_in`: int
  - Number of features seen during fitting the model
- `support_`: ndarray of shape `(n_SV)`
  - Indices of support vectors.
- `support_vectors_`: ndarray of shape `(n_SV, n_features)`
  - Support vectors.

# Kernel SVM



- An example: based on observations of dosage of a medicine, determine if a dosage is effective or not

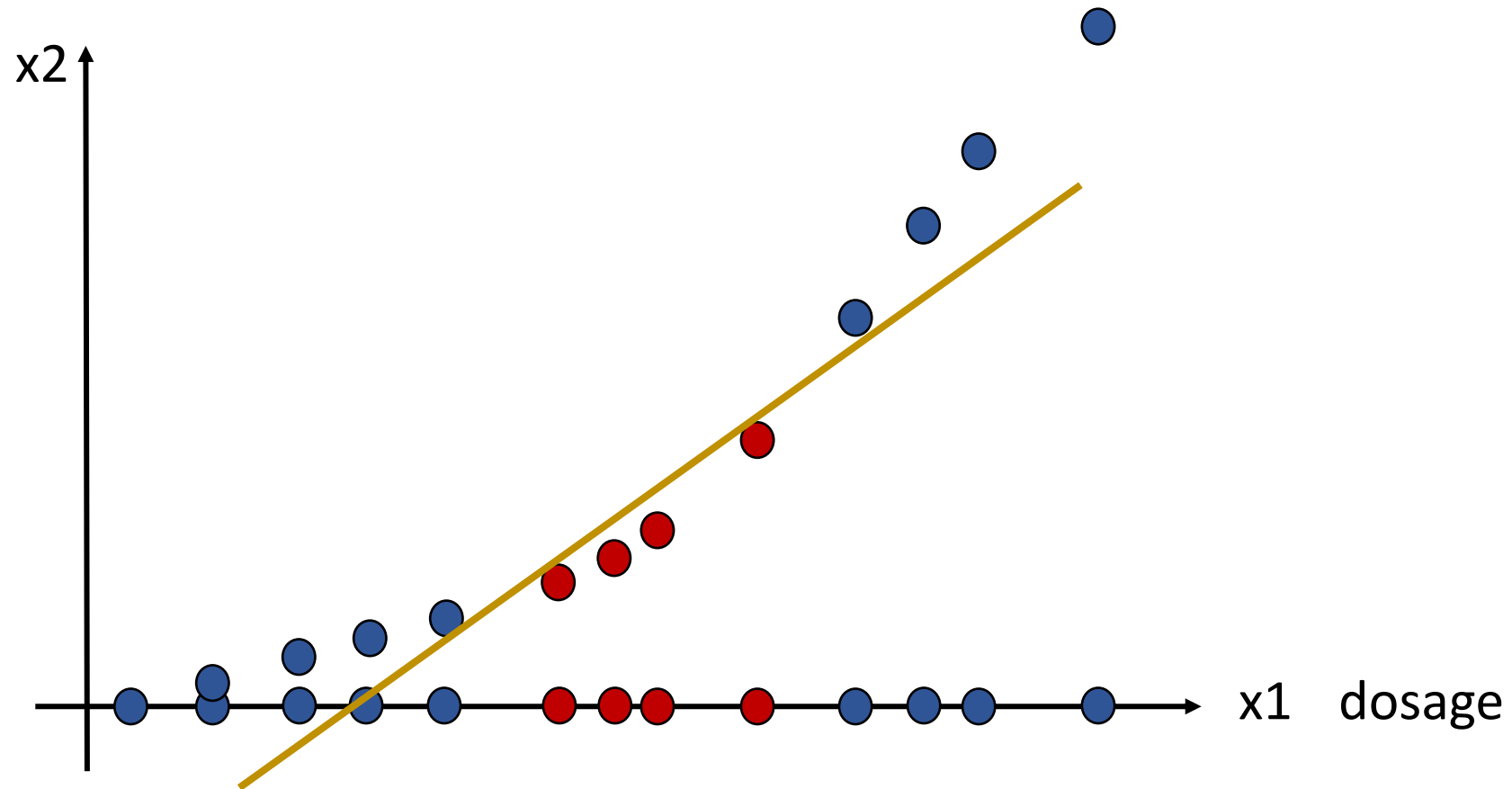




# Kernel SVM



- Transforming features to higher dimensions





# Kernel SVM

- Recall the SVM solution for weights:

$$\mathbf{w} = \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i$$

- For an input  $\mathbf{x}$ :

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + b$$



# Kernel SVM

- Assume we transfer  $\mathbf{x}$  to  $\Phi(\mathbf{x})$ , we have

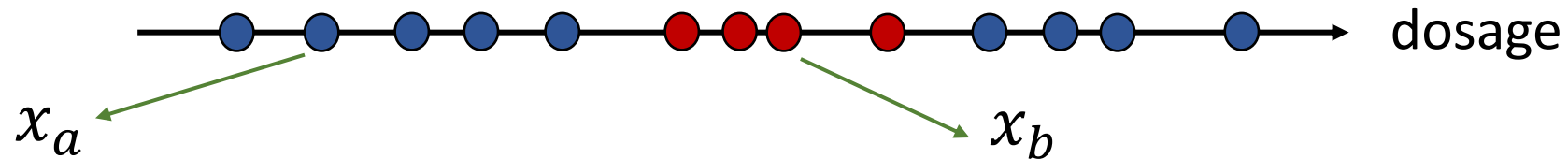
$$f(\Phi(\mathbf{x})) = \sum_{i=1}^n \lambda_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + b$$

Kernel, also can be written as  $K(\mathbf{x}_i, \mathbf{x})$



# Polynomial Kernel (Optional)

- A simple polynomial kernel in 1D dimension:

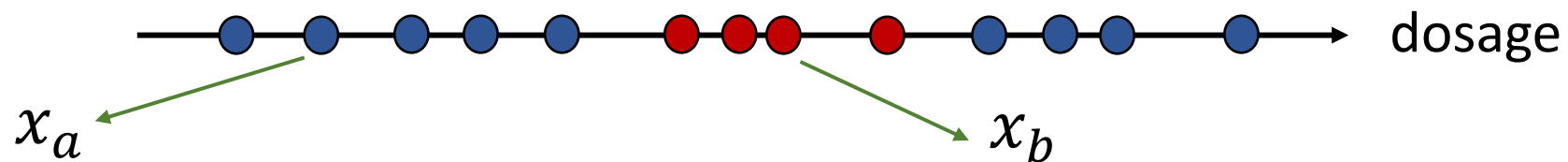


$$K(x_a, x_b) = \left(x_a x_b + \frac{1}{2}\right)^2$$

# Polynomial Kernel (Optional)



$$\begin{aligned} K(x_a, x_b) &= \left(x_a x_b + \frac{1}{2}\right)^2 = x_a x_b + x_a^2 x_b^2 + \frac{1}{4} \\ &= \left\{x_a, x_a^2, \frac{1}{2}\right\}^T \left\{x_b, x_b^2, \frac{1}{2}\right\} \end{aligned}$$

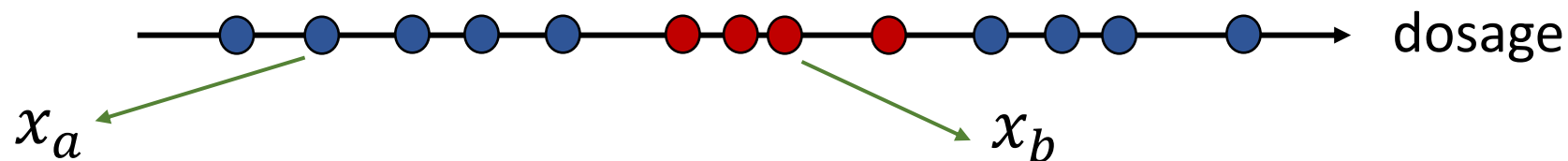




# Polynomial Kernel (Optional)

- A general polynomial kernel in multi-feature dimension:

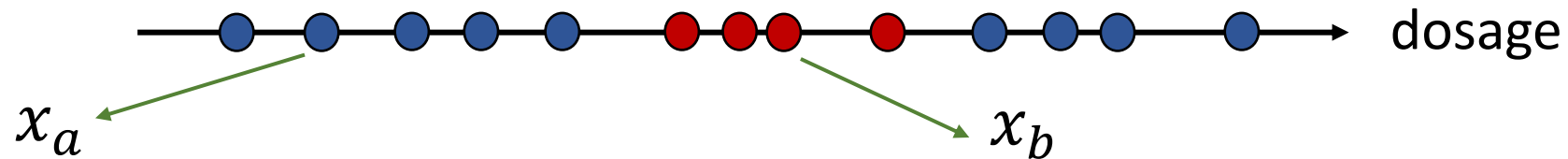
$$K(\mathbf{x}_a, \mathbf{x}_b) = (\mathbf{x}_a^T \mathbf{x}_b + r)^d$$





# RBF Kernel (Optional)

- A simple RBF kernel in 1D dimension:



$$K(x_a, x_b) = e^{-\frac{1}{2}(x_a - x_b)^2}$$

# RBF Kernel (Optional)



- RBF naturally contains a polynomial kernel in infinite space

$$\begin{aligned} K(x_a, x_b) &= e^{-\frac{1}{2}(x_a - x_b)^2} = e^{-\frac{1}{2}(x_a^2 + x_b^2) + x_a x_b} \\ &= e^{-\frac{1}{2}(x_a^2 + x_b^2)} e^{x_a x_b} \end{aligned}$$



# RBF Kernel (Optional)



- RBF naturally contains a polynomial kernel in infinite space

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^\infty}{\infty!}$$

$$e^{x_a x_b} = 1 + x_a x_b + \frac{(x_a x_b)^2}{2!} + \frac{(x_a x_b)^3}{3!} + \dots + \frac{(x_a x_b)^\infty}{\infty!}$$

$$= \left\{ 1, x_a, \frac{x_a^2}{2!}, \frac{x_a^3}{3!}, \dots, \frac{x_a^\infty}{\infty!} \right\}^T \left\{ 1, x_b, \frac{x_b^2}{2!}, \frac{x_b^3}{3!}, \dots, \frac{x_b^\infty}{\infty!} \right\}$$



# RBF Kernel (Optional)

- RBF naturally contains a polynomial kernel in infinite space

$$K(x_a, x_b) = e^{-\frac{1}{2}(x_a^2 + x_b^2)} e^{x_a x_b}$$
$$= e^{-\frac{1}{2}(x_a^2 + x_b^2)} \left\{ 1, x_a, \frac{x_a^2}{2!}, \frac{x_a^3}{3!}, \dots, \frac{x_a^\infty}{\infty!} \right\}^T \left\{ 1, x_b, \frac{x_b^2}{2!}, \frac{x_b^3}{3!}, \dots, \frac{x_b^\infty}{\infty!} \right\}$$

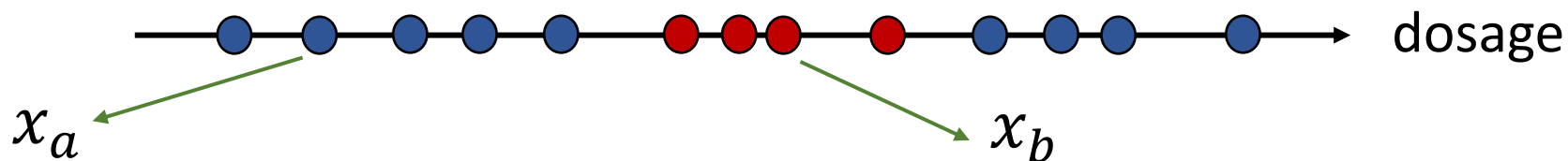


# RBF Kernel (Optional)

- A general RBF kernel in multi-feature dimension:

$$K(\mathbf{x}_a, \mathbf{x}_b) = e^{-\frac{1}{\sigma^2}(\|\mathbf{x}_a - \mathbf{x}_b\|)^2}$$

- It measures the influence one sample has over another sample



# Kernel SVM



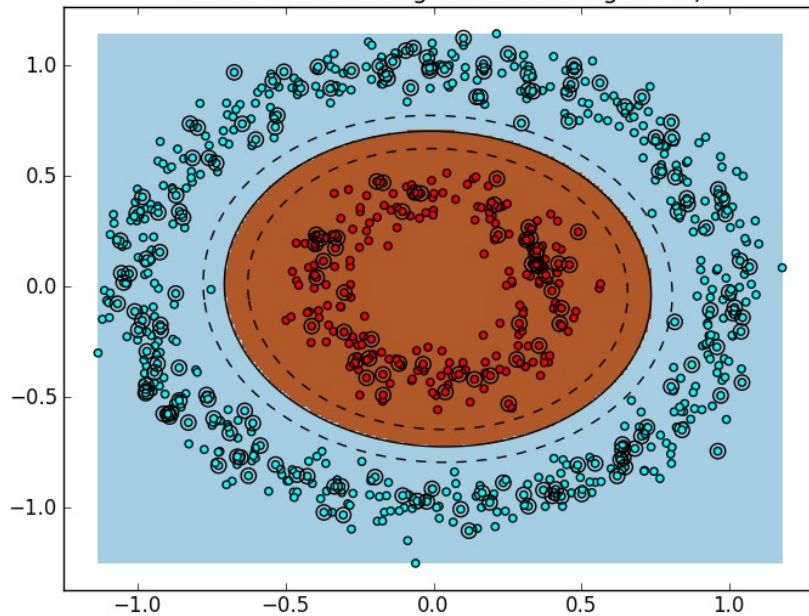
- We don't actually conduct feature transformation, i.e.,  $x$  to  $\Phi(x)$
- Instead, we apply kernel trick to obtain the dot product of the transformed features in high dimensional space

# Kernel SVM

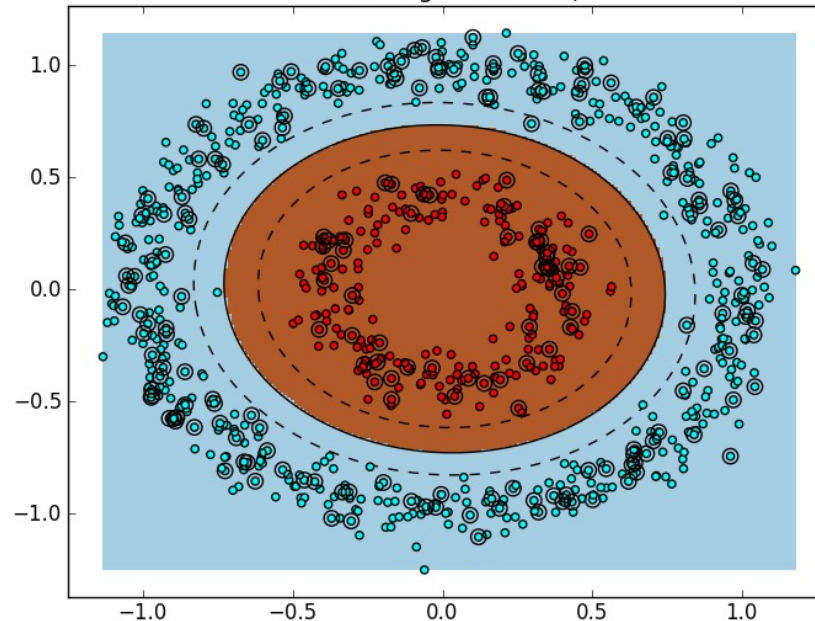


- Kernel options provided by Scikit-Learn: 'linear', '**poly**', '**rbf**', 'sigmoid', 'precomputed'

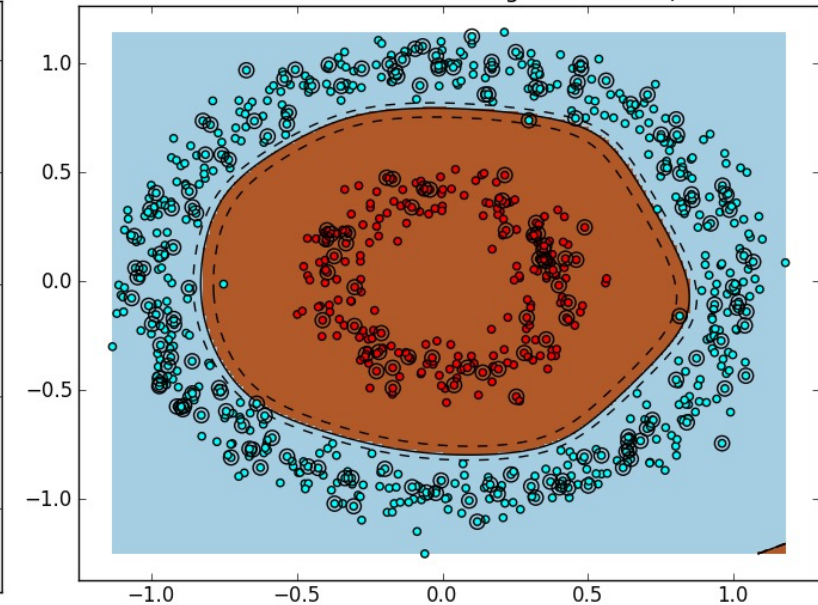
SVM Decision Boundary accuracy=1.0 (Kernel=poly  
C=1.0 coef0=10.0 gamma=0.1 degree=4)



SVM Decision Boundary accuracy=1.0 (Kernel=rbf  
C=10.0 gamma=0.1)



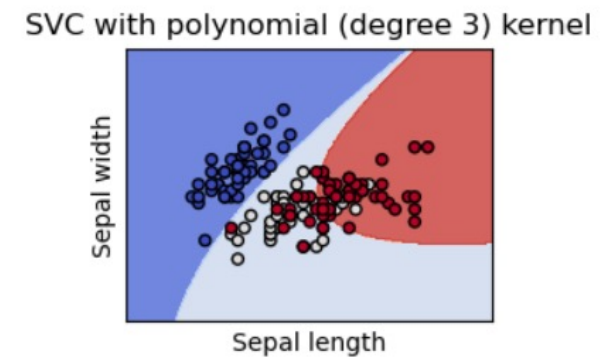
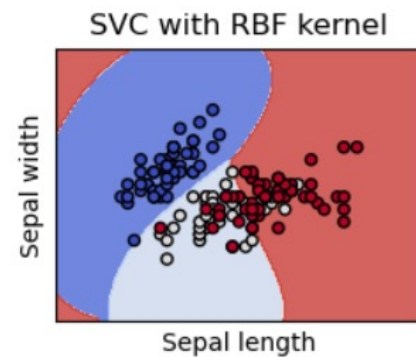
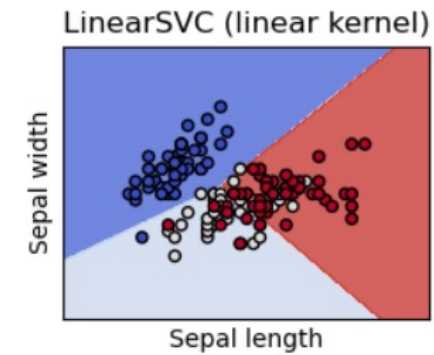
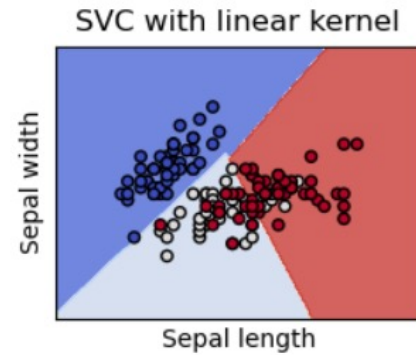
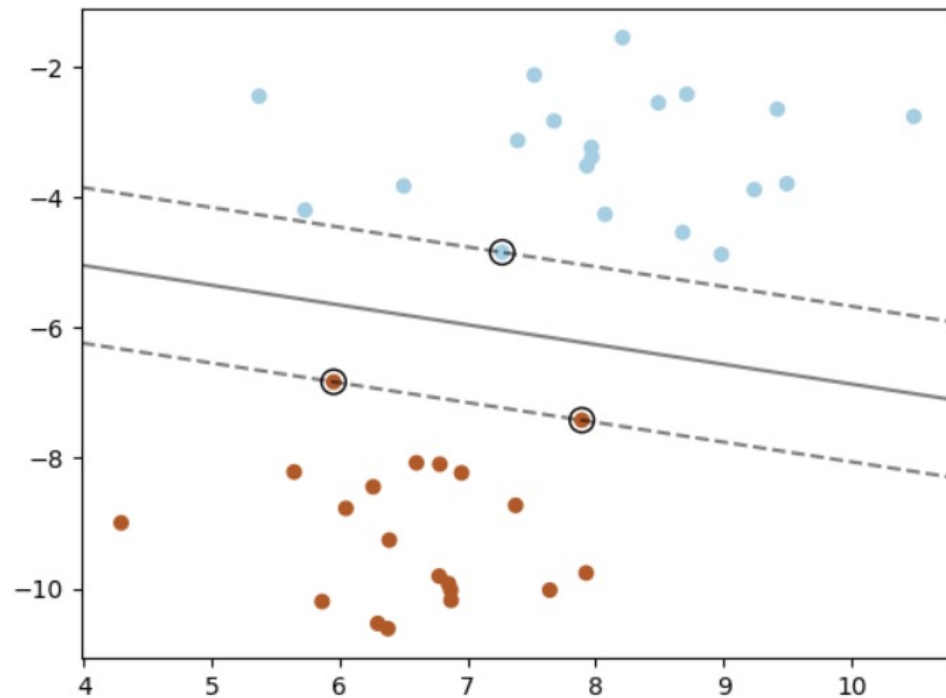
SVM Decision Boundary accuracy=0.99 (Kernel=sigmoid  
C=1000.0 coef0=-10.0 gamma=10.0)





# Two SVC Demos

- (1) Maximum margin separating hyperplane
- (2) Plot different kernels using iris dataset





# A2: Point Cloud Classification

- You will use a classical ML model to perform point cloud classification (again, on object level)
- You are allowed to use libraries such as scikit learn
- You will need to evaluate the performance of the classifier



# A2: Evaluation of the Model

- **OA**: overall accuracy
  - Out of 500 objects, how many are correctly classified?
- **mAcc**: mean per-class accuracy
  - How is the accuracy of each class? Average them.





# A2: Evaluation of the Model

- **Confusion Matrix:**

- Row is the actual class, while column is the predicted class

	Count	Building	Car	Fence	Pole	Tree	
							← True classes
	Building						
Pred. classes →	Car						
	Fence						
	Pole						
	Tree						



Questions?