Chapter 9(3rd ed)/7(4th ed). Classification: Advanced Methods

- Feature Selection
- Bayesian Belief Networks
- Support Vector Machines
- Rule-Based and Pattern-Based Classification
- Weakly Supervised Learning
- Classification with Rich Data Type
- Other Related Techniques
- Summary

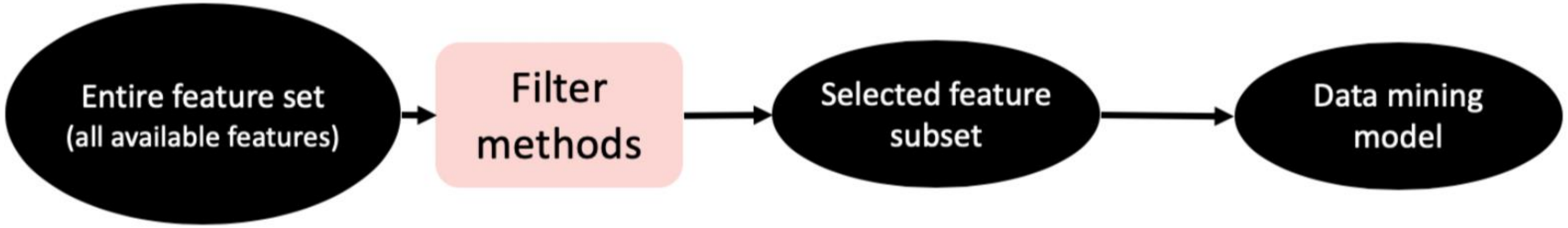
Feature Selection & Feature Engineering

- **Feature Selection**
  - Given a set of *p* initial features, how to select a few most effective ones?
  - Why?
    - Irrelevant features (student ID for predicting GPA)
    - Redundant features (monthly income vs. yearly income)
- **Feature Engineering**
  - Given the initial features, how to construct more effective ones?
    - # of daily positive cases, # of daily tests, # of daily hospitalization → weekly positive rate
  - (traditionally) domain knowledge is the key
  - Deep learning provides an automatic way

Feature Selection Methods

- **Filter methods**
  - Select features based on some goodness measure
  - Independent of the specific classification model
- **Wrapper methods**
  - Combine the feature selection and classifier model construction steps together,
  - Iteratively
    - Use the currently selected feature subset to construct a classification model
    - Use the current classification model to update the selected feature subset.
- **Embedded methods**
  - Simultaneously constructs the classification model and selects the relevant features
  - Embed the feature selection step during the classification model construction step

Filter Methods

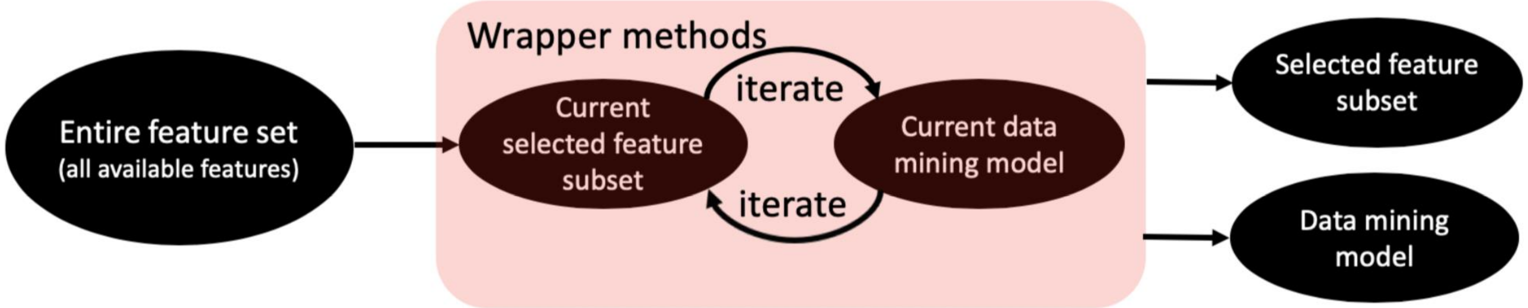


- **General Procedure**
  - Selects features based on some goodness measure
  - Independent of the specific classification model
- **Fisher Scores**
  - **Intuitions**: the feature x (e.g., income) is strongly correlated with the class label y (buy computer) if
    - the average income of all customers who buy a computer is significantly different from the average income of all customers who do not buy a computer,
    - all customers who buy a computer share similar income, and
    - all customers who do not buy a computer share similar income.
  - **Details**

$$s = \frac{\sum_{j=1}^{c} n_j (\mu_j - \mu)^2}{\sum_{j=1}^{c} n_j \sigma_j^2}$$

- **Other goodness measures**
  - χ^2 test (for categorical feature); information gain, mutual information.

Wrapper Methods



- **General Procedure**
  - Combines the feature selection and classifier model construction steps together,
  - Iteratively
    - Use the currently selected feature subset to construct a classification model
    - Use the current classification model to update the selected feature subset.
- **Key: how to search for the best feature subset**
  - Exhaustive search: 2^p-1 (try all the possible subsets of the *p* given features)
  - Stepwise forward selection:
    - Start with an empty feature subset.
    - At each iteration, select an additional feature to improve performance most
  - Stepwise backward elimination: start with the full set, eliminate one feature at a time
  - Hybrid method

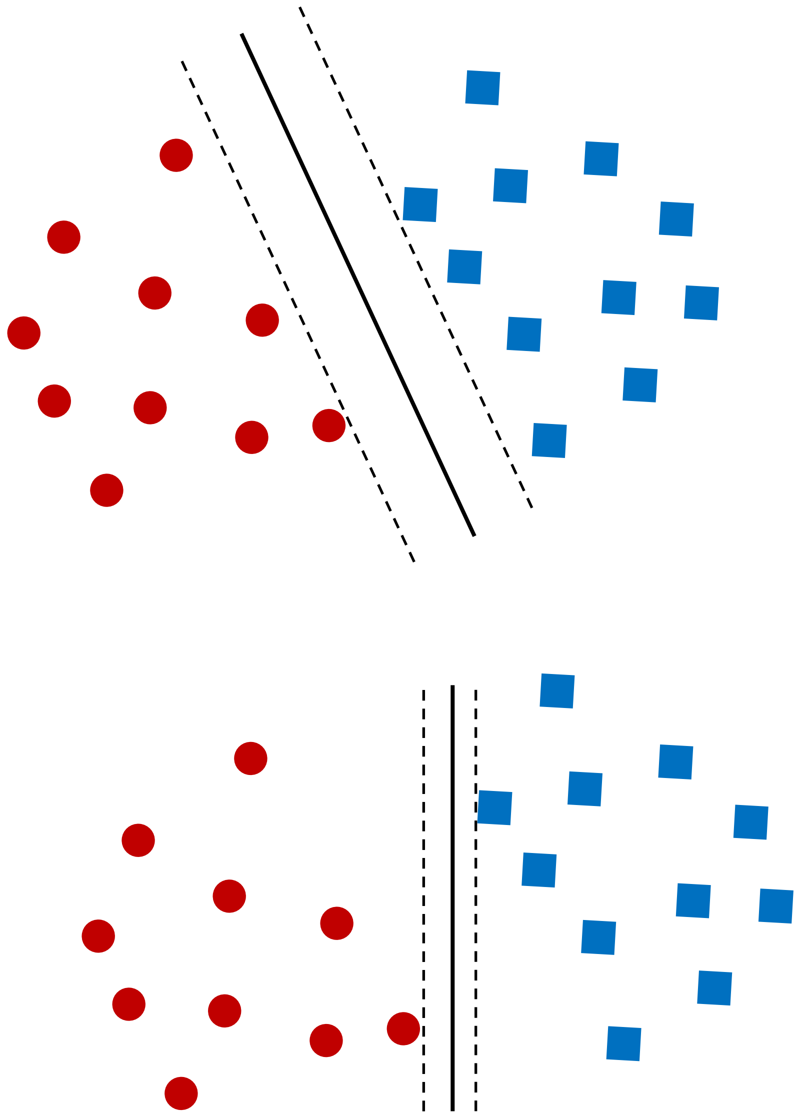
Chapter 9(3rd ed)/7(4th ed). Classification: Advanced Methods

- Feature Selection
- Bayesian Belief Networks
- Support Vector Machines
- Rule-Based and Pattern-Based Classification
- Weakly Supervised Learning
- Classification with Rich Data Type
- Other Related Techniques
- Summary

Classification: A Mathematical Mapping

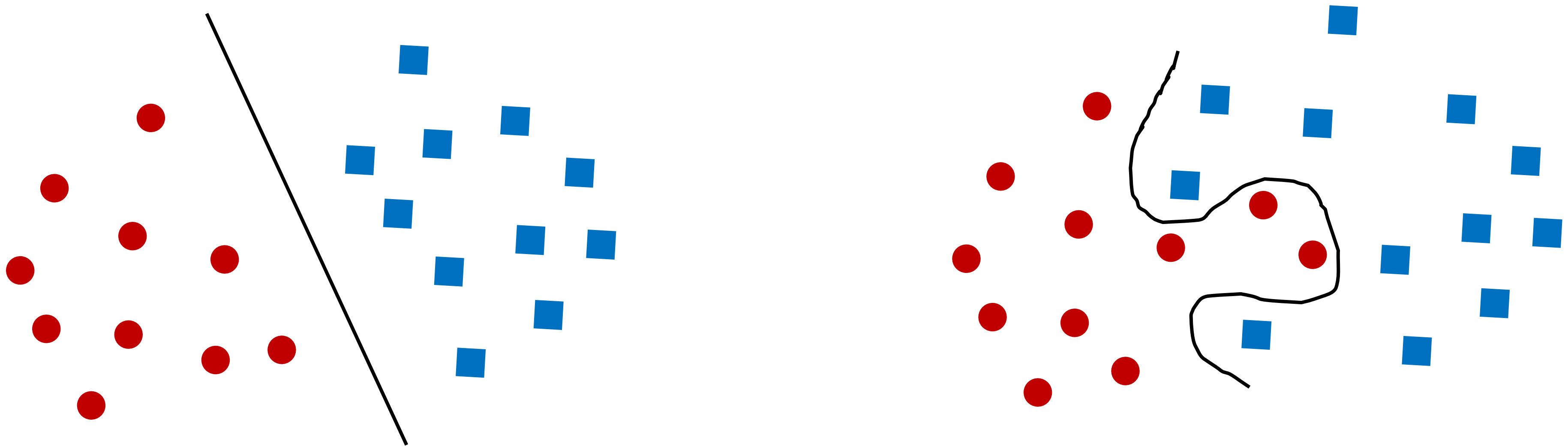
- **The binary classification problem:**
  - E.g., Movie review classification
    - $x_i = (x_1, x_2, x_3, \ldots)$, $y_i = +1$ or -1 (positive, negative)
    - x_1 : # of word "awesome"
    - x_2 : # of word "disappointing"
- Mathematically, $\boldsymbol{x} \in X = \Re^n, y \in Y = \{+1, -1\}$
  - We want to derive a function $f\colon\ X \rightarrow\ Y$
  - which maps input examples to their correct labels

SVM—General Philosophy



- Learning a max-margin classifier
  - From the infinite set of lines (hyperplanes) separating two classes
  - Find the one which separates two classes with the **largest margin**
  - i.e. a **maximum marginal hyperplane (MMH)**

SVM—When Data Is Linearly Separable



Linearly Separable   Linearly Inseparable

- The simplest case: When data is **linearly separable**
  - Data sets whose classes can be separated exactly by linear decision surfaces are said to be linearly separable

Linear SVM for Linearly Separable Data

- A separating hyperplane can be written as

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

Model parameters to learn

where $w = (w_1, w_2, \dots, w_n)^T$ is a weight vector and b a scalar (bias)

  - For 2-D, it can be written as: $w_1 x_1 + w_2 x_2 + b = 0$

- The hyperplane defining the sides of the margin:

  H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$

- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**

Linear SVM for Linearly Separable Data

- The distance from any data point $\boldsymbol{x}$ to the separating hyperplane is

$$\text{distance}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} \qquad r = \frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|} = \frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|}$$

- Our objective is to maximize the distance of the closest data point to the hyperplane

$$\arg\max_{w,b}\left\{\frac{1}{\|\boldsymbol{w}\|}\min[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)]\right\}$$

- This is hard to solve, we shall convert it to an easier problem

$$\boxed{\begin{aligned} \arg\min_{w,b} \quad & \|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \qquad i = 1,2,\dots,n \end{aligned}}$$

- This is the basic form of SVM, and it can be solved by using *quadratic programming*

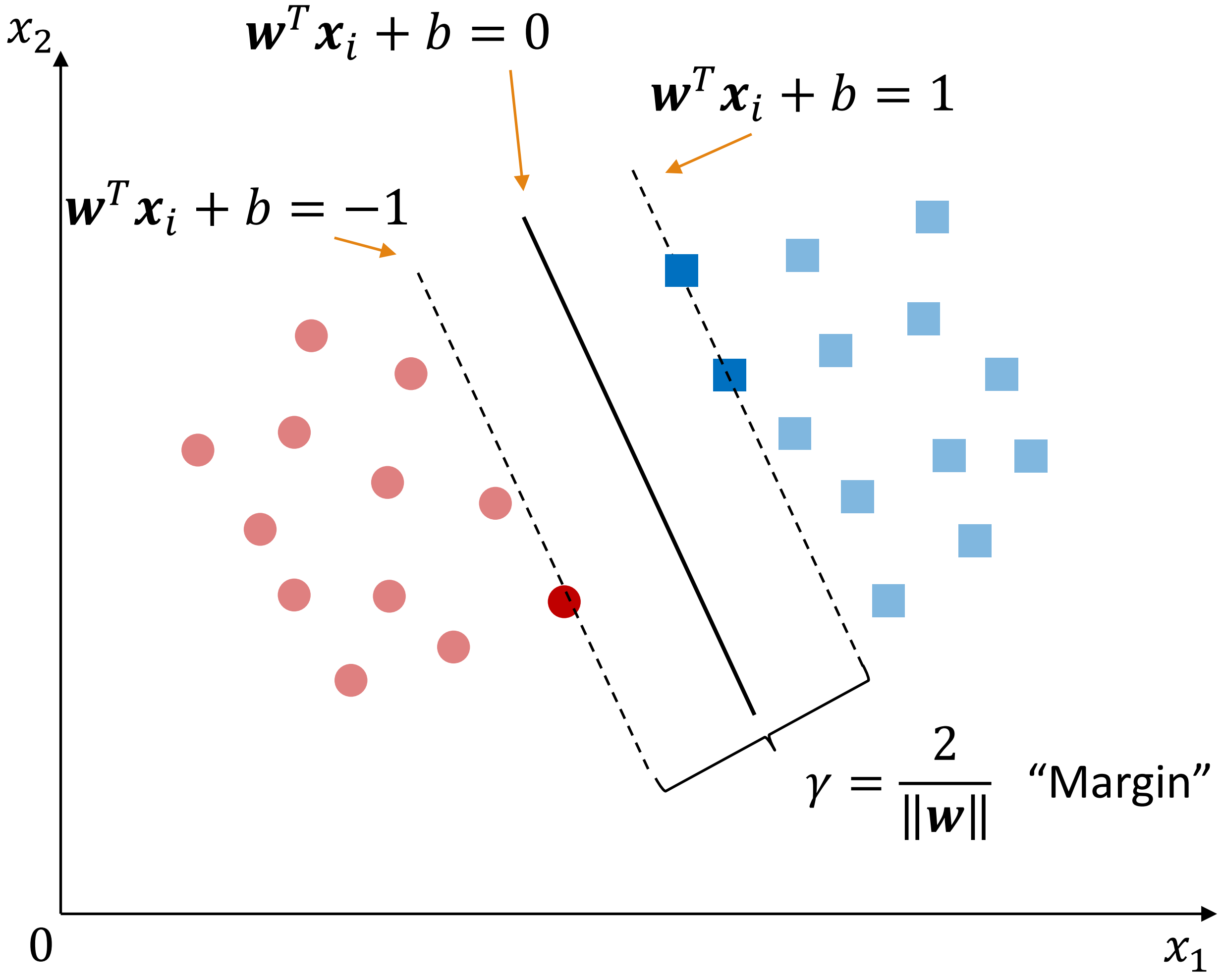
Linear SVM for Linearly Separable Data

- *"Once I've got a trained support vector machine, how do I use it to classify test (i.e., new) tuples?"*
- Based on the Lagrangian formulation mentioned before, the MMH can be rewritten as the decision boundary

$$d(X) = \sum_{i=1}^{l} y_i \alpha_i X' X_i + b,$$

- where y_i is the class label of support vector $\boldsymbol{X_i}$; $\boldsymbol{X}$ is a test tuple and denotes the transpose of a vector; α_i and b are numeric parameters that were determined automatically by the optimization or SVM algorithm noted before; and l is the number of support vectors, which is often much smaller than the total number of training tuples

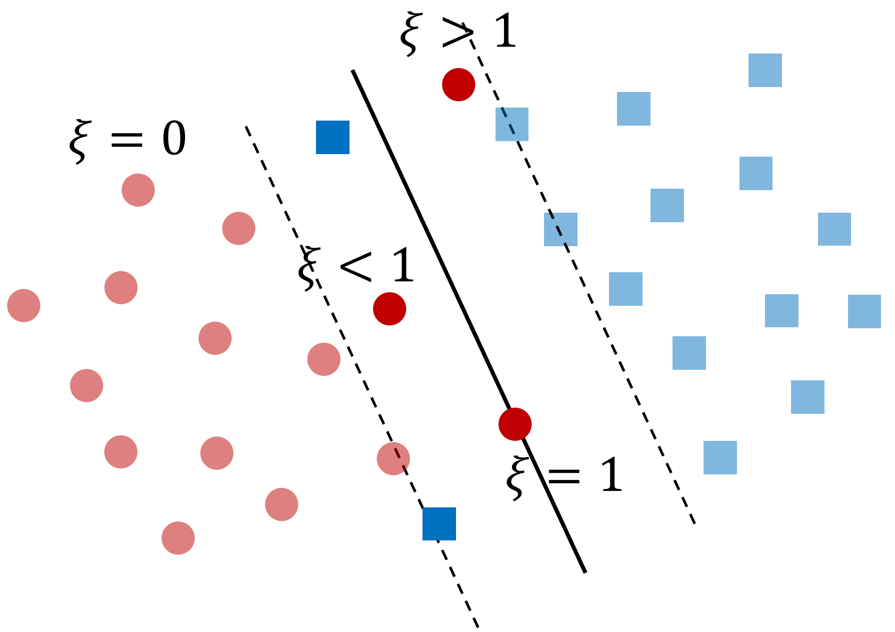
Linear SVM for Linearly Separable Data



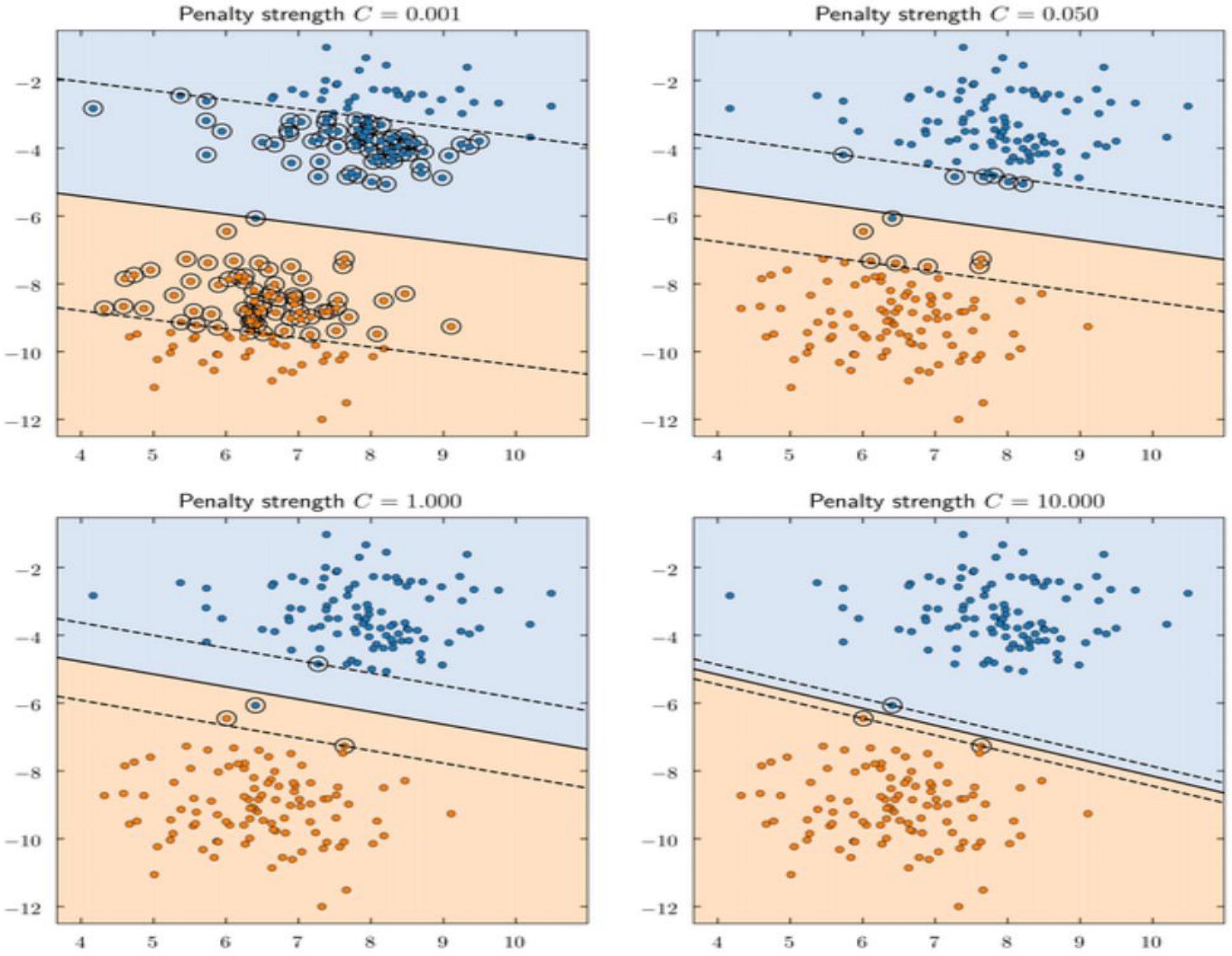
- The data points closest to the separating hyperplane are called **support vectors**

SVM for Linearly Inseparable Data

- We allow data points to be on the "wrong side" of the **margin boundary**
- Penalize points on the wrong side according to its distance to the margin boundary
- ξ : slack variable
- C (> 0): Controls the trade-off between the penalty and the margin
- Smaller C: allow more mistake
- Larger C: allow less mistake
- This is the widely used *soft-margin SVM*



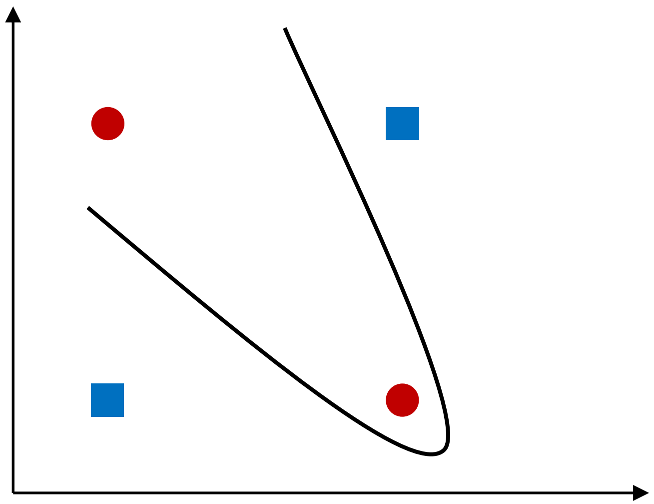
Effect of slack variable



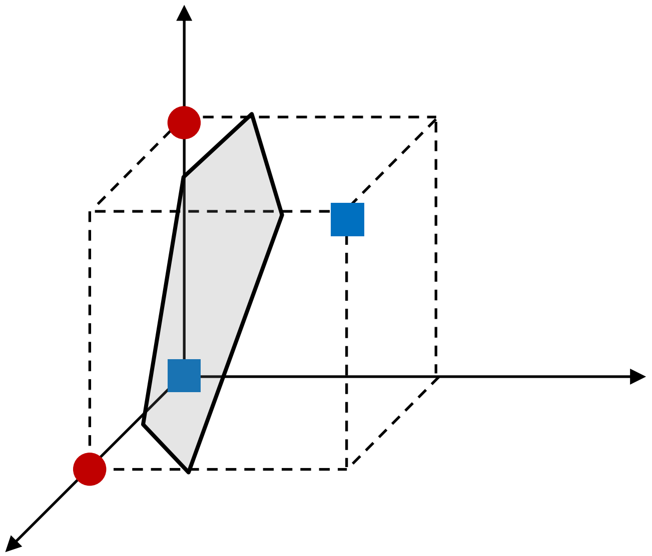
https://www.quora.com/What-is-the-purpose-for-using-slack-variable-in-SVM

SVM for Linearly Inseparable Data

- Alternatively, for linearly inseparable data, we can map them to a higher dimensional space
- We search for a linear separating hyperplane in the new space
  - Example: The XOR problem



$\boldsymbol{x} \longmapsto \phi(\boldsymbol{x})$



Kernel Functions for Nonlinear Classification

- Instead of computing the dot product on the transformed data, it is mathematically equivalent to applying a kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to the original data, i.e.,
  - $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)\,\phi(\boldsymbol{x}_j)$
- Typical Kernel Functions

$$\textbf{Polynomial kernel of degree } h: \quad K(X_i, X_j) = (X_i \cdot X_j + 1)^h$$

$$\textbf{Gaussian radial basis function kernel}: \quad K(X_i, X_j) = e^{-\|X_i - X_j\|^2/2\sigma^2}$$

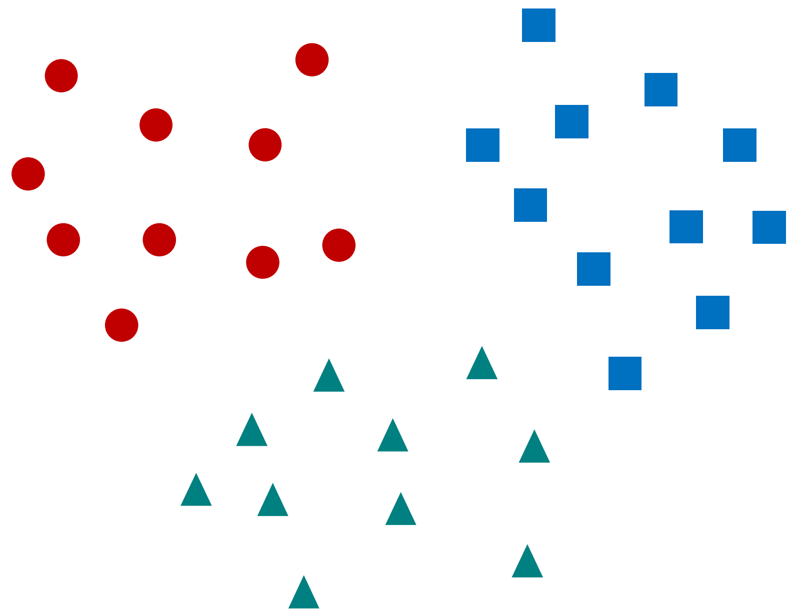
$$\textbf{Sigmoid kernel}: \quad K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$$

- SVMs can efficiently perform a non-linear classification using kernel functions, implicitly mapping their inputs into high-dimensional feature spaces

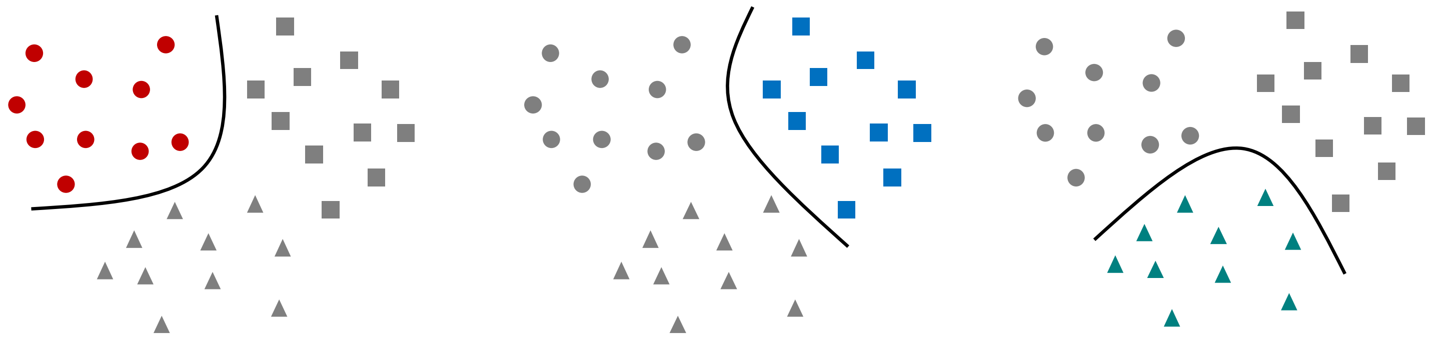
https://www.youtube.com/watch?time_continue=42&v=3liCbRZPrZA

http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/

Multi-class Classification with SVM

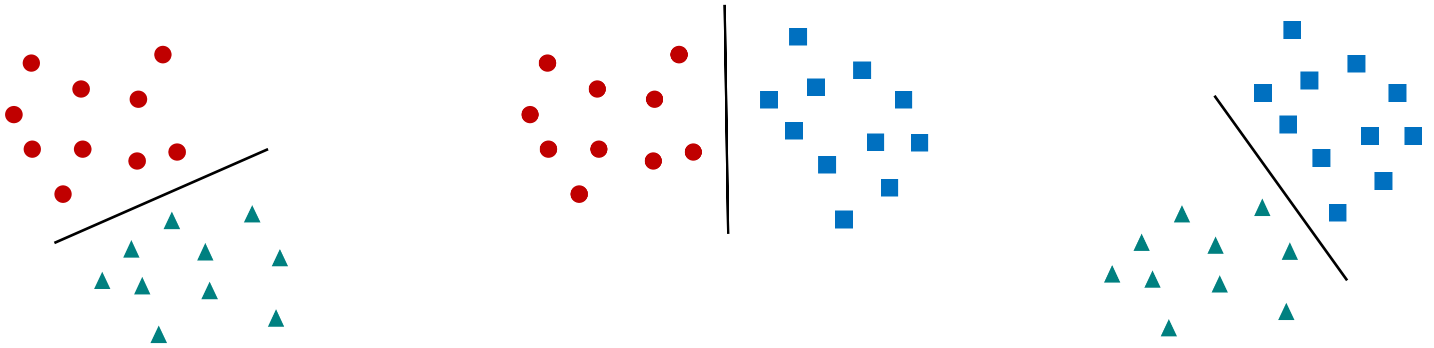


N classes



One-vs-Rest

Requires N classifiers



One-vs-One

Requires N(N - 1)/2 classifiers

Is SVM Scalable on Massive Data?

- SVM is effective on high dimensional data
  - The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
  - The **support vectors** are the essential or critical training examples—they lie closest to the decision boundary (MMH)
  - Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high
- SVM is not scalable to the # of data objects in terms of training time and memory usage
  - Scaling SVM by a hierarchical micro-clustering approach
    - H. Yu, J. Yang, and J. Han, "Classifying Large Data Sets Using SVM with Hierarchical Clusters", KDD'03

SVM: Applications

- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used for: classification and numeric prediction
  - SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters)
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

SVM Recap

- Pros
  - Elegant mathematical formulation, guaranteed global optimal with optimization
  - Trains well on small data sets
  - Flexibility through kernel functions
  - Conformity with semi-supervised training
- Cons
  - Not naturally scalable to large data sets

SVM Related Links

- SVM Website: http://www.kernel-machines.org/
- Representative implementations
  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - **SVM-light**: simpler but performance is not better than LIBSVM, support only binary classification and only in C
  - **SVM-torch**: another recent implementation also written in C

SVM Related Links

- SVM Website: http://www.kernel-machines.org/
- Representative implementations
  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - **SVM-light**: simpler but performance is not better than LIBSVM, support only binary classification and only in C
  - **SVM-torch**: another recent implementation also written in C