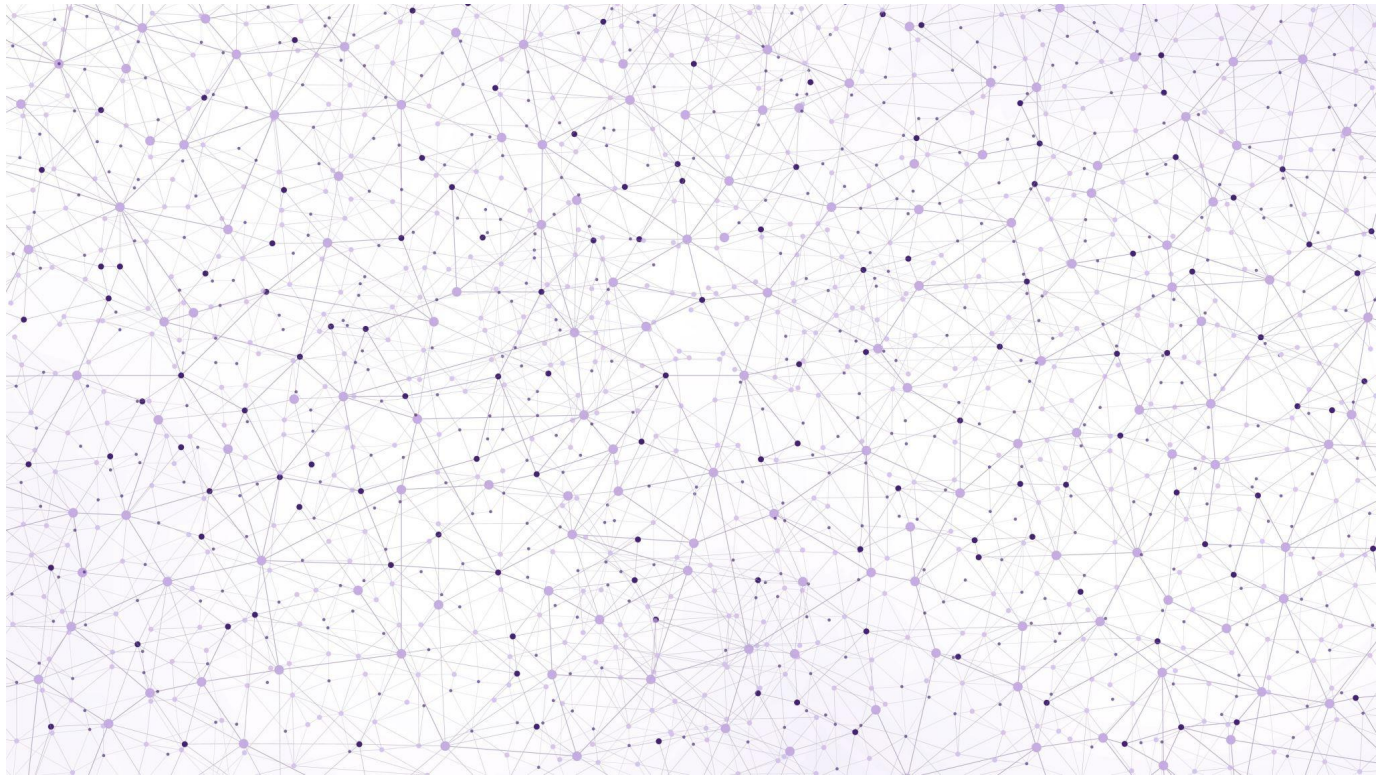
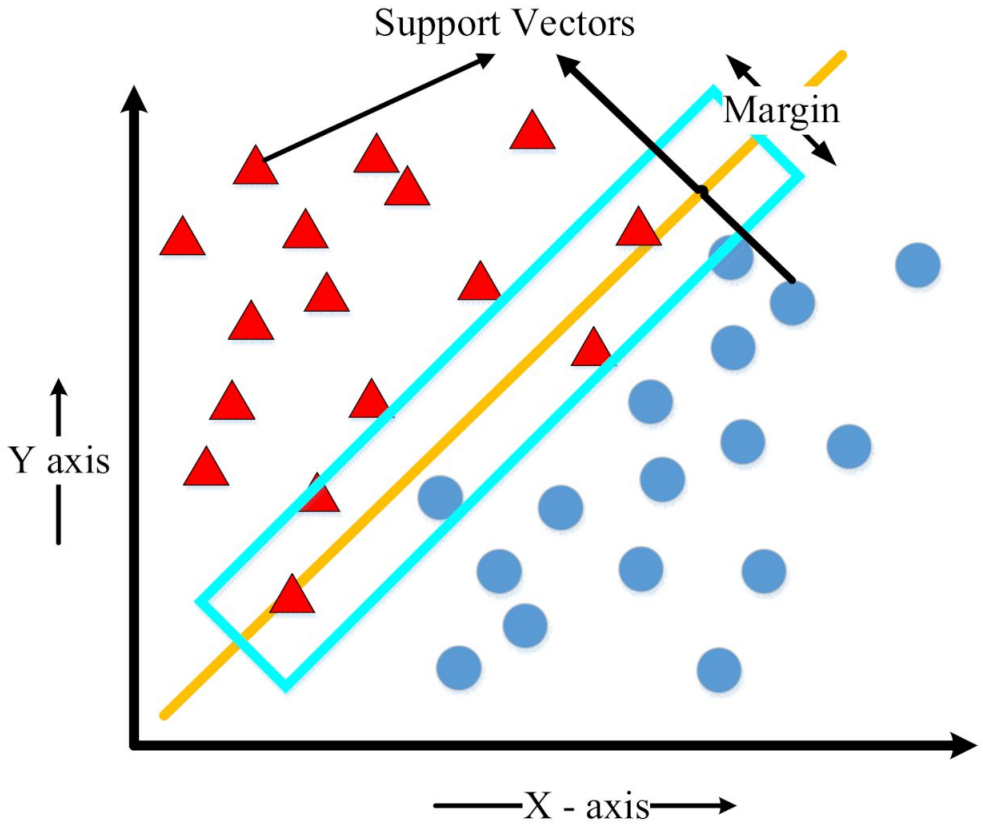
Support Vector Machine

Dr. Risman Adnan Mattotorang

Telkom University





Outline

- Vector and Hyperplane
- Main Ideas of SVM
- SVM Optimization Problem
- Non-Linear SVMs
- Introduction to Kernels

Vector and Hyperplane

– Vector and Hyperplane

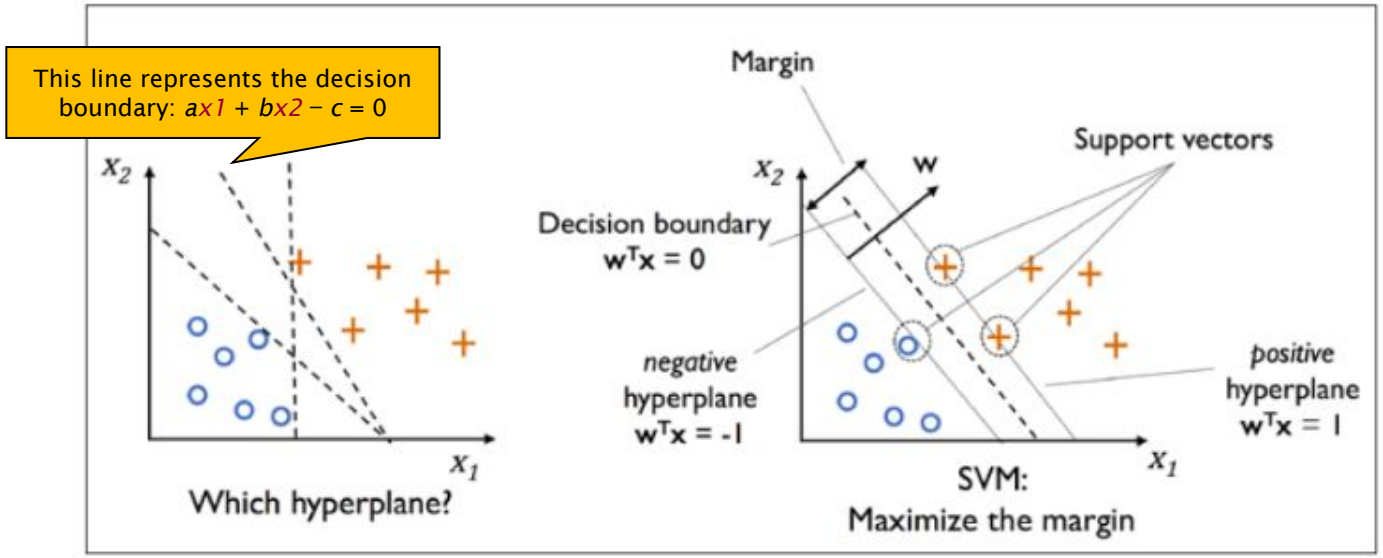
Lots of possible solutions for *a, b, c.*

Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]

E.g., perceptron

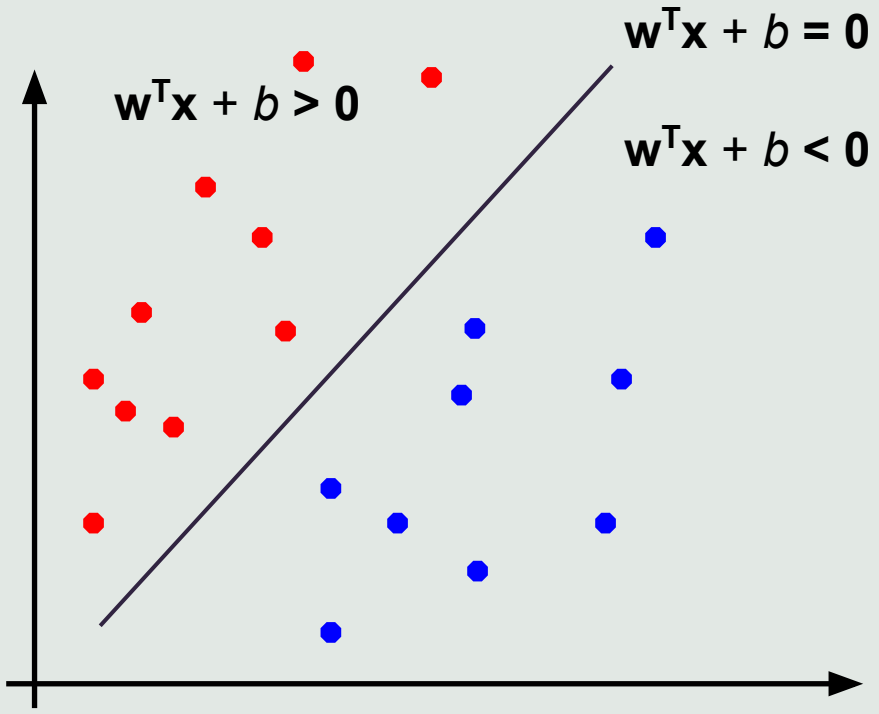
Support Vector Machine (SVM) finds an optimal* solution.

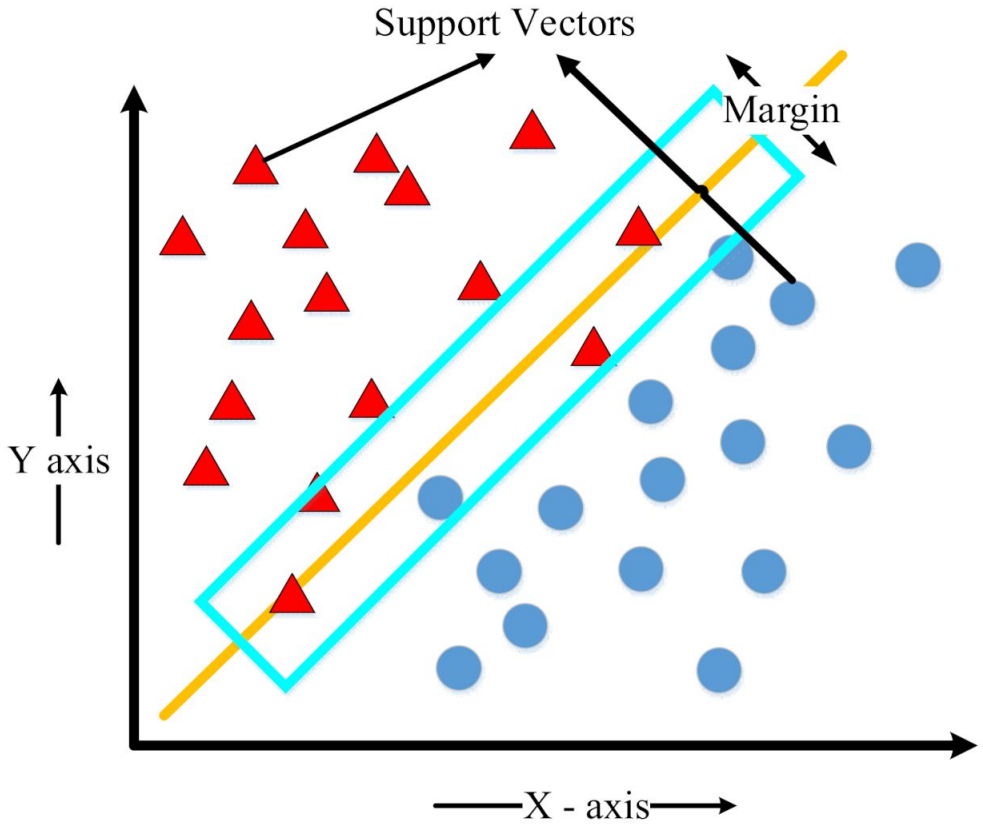
Maximizes the distance between the hyperplane and the “difficult points” close to decision boundary
One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions



Perceptron Revisited

- Perceptron is Linear Separator
- Binary classification can be viewed as the task of separating classes in feature space



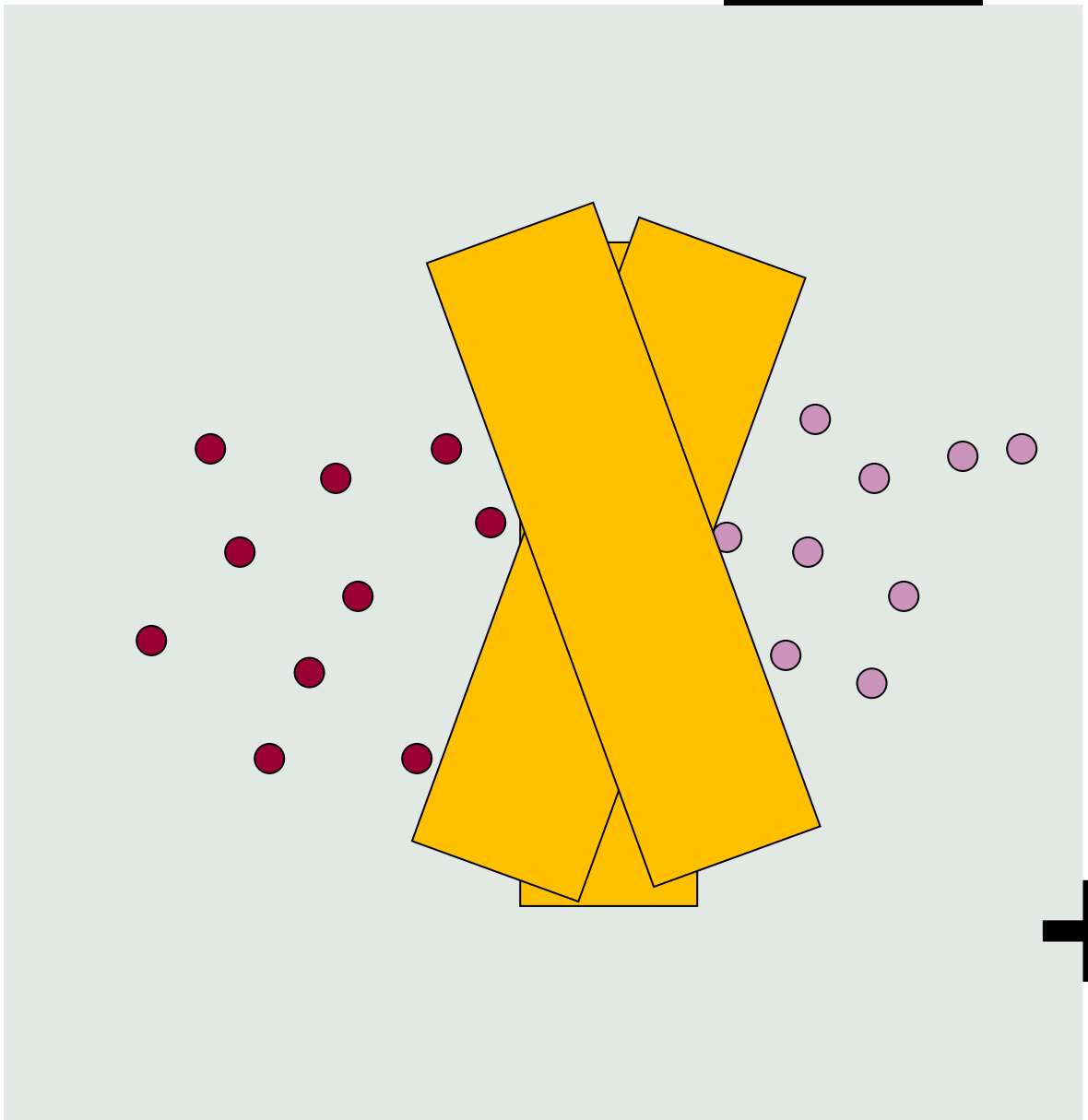


Outline

- Vector and Hyperplane
- Main Ideas of SVM
- SVM Optimization Problem
- Non-Linear SVMs
- Introduction to Kernels

Intuition of SVM

– If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



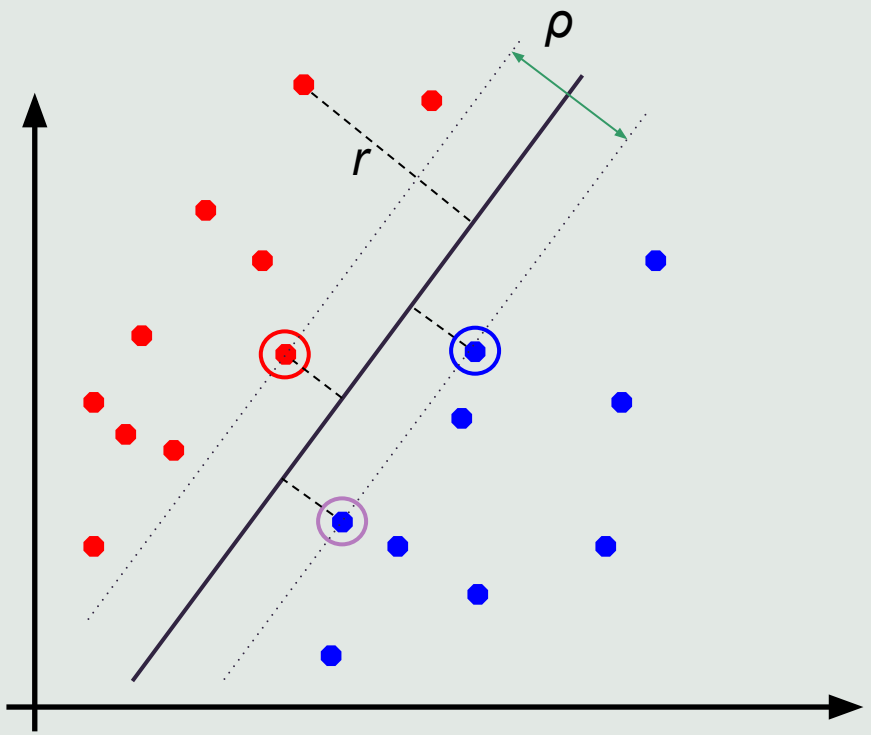
Geometric Margin

– Distance from example $\mathbf{x}_i$ to the separator is:

$$r = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$$

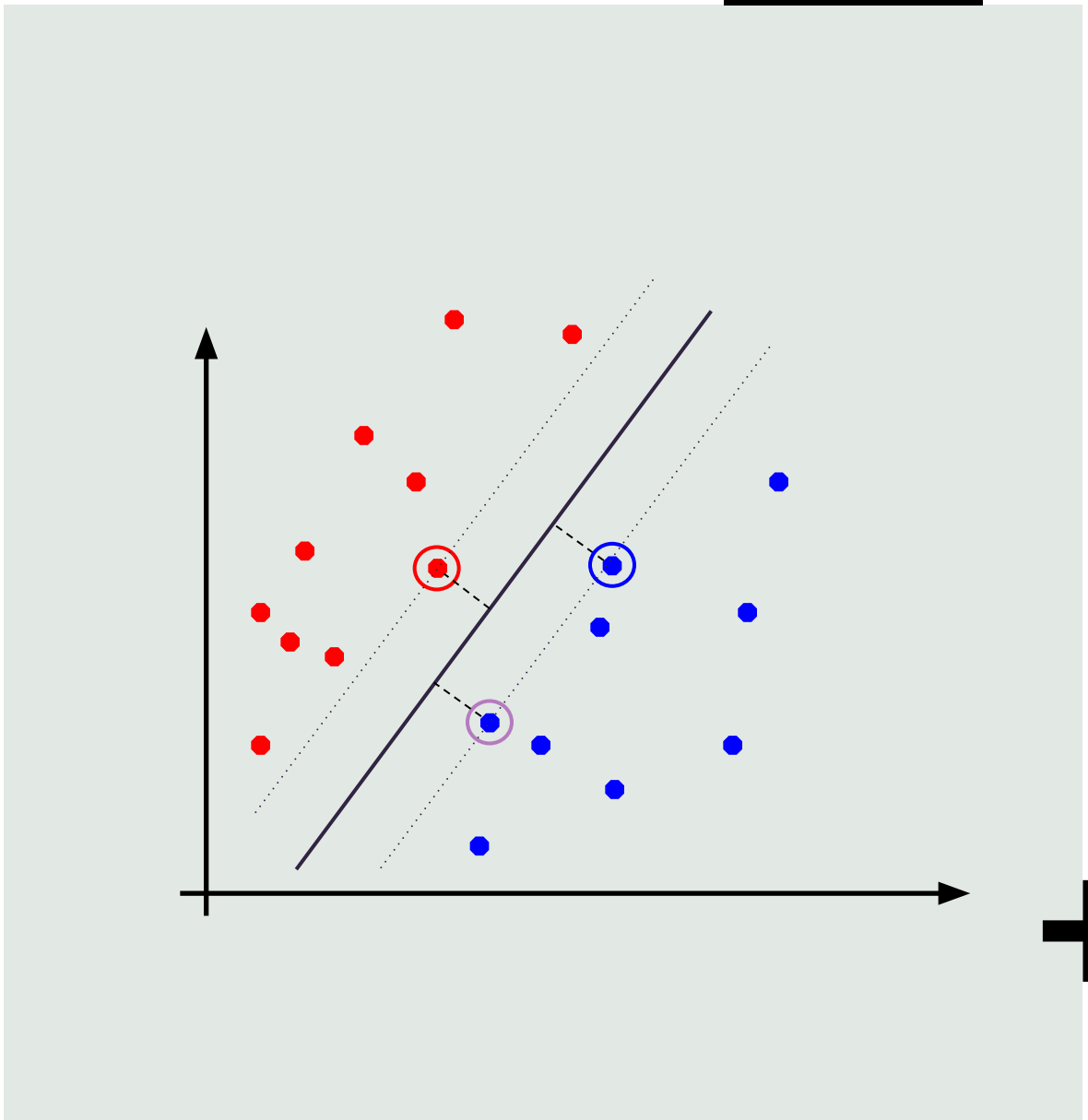
– Examples closest to the hyperplane are ***support vectors***.

– ***Margin*** ρ of the separator is the distance between support vectors.



Maximum Margin Classification

- Maximizing the margin is good according to intuition and PAC theory.
- Implies that only support vectors matter; other training examples are ignorable.



Support Vector Machine (SVM)

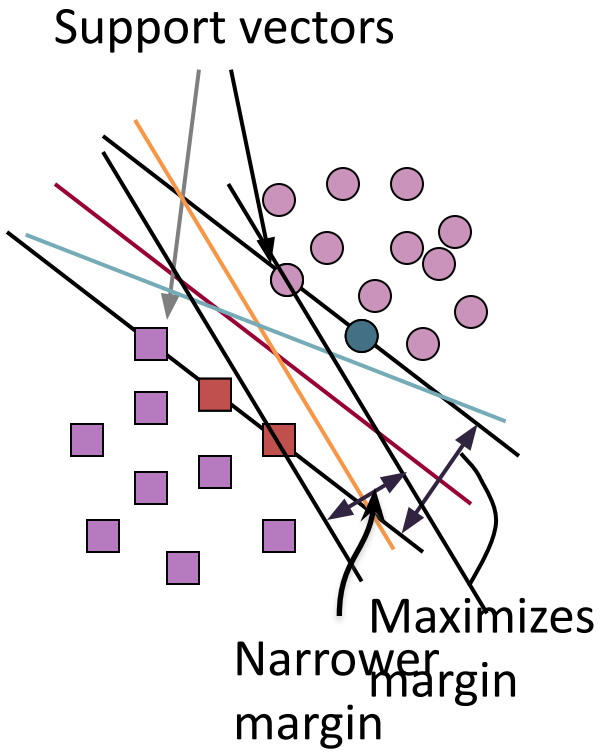
SVMs maximize the *margin* around the separating hyperplane.

A.k.a. large margin classifiers

The decision function is fully specified by a subset of training samples, *the support vectors*.

Solving SVMs is a *quadratic programming* problem

Seen by many as the most successful current text classification method*



*but other discriminative methods often perform very similarly

Linear SVM Mathematically

The linearly separable case

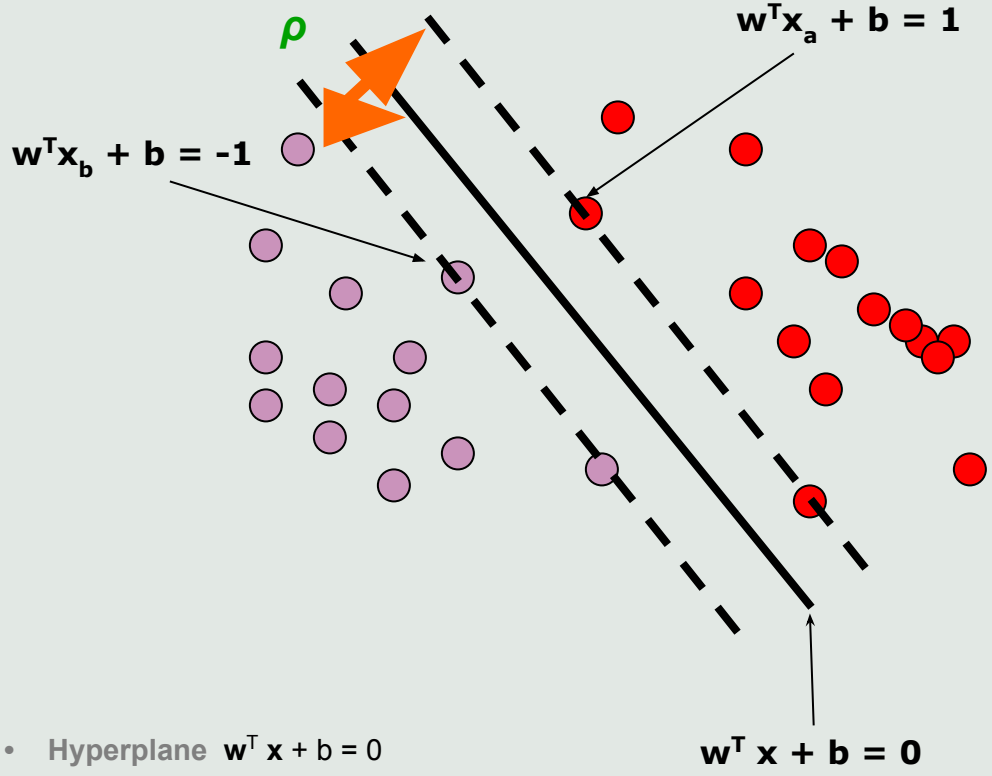
– Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w}^T\mathbf{x_i} + b \geq 1 \quad \text{if } y_i = 1$$
$$\mathbf{w}^T\mathbf{x_i} + b \leq -1 \quad \text{if } y_i = -1$$

– For support vectors, the inequality becomes an equality. Then, since each example's distance from the hyperplane is:

$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

– The margin is: $\rho = \frac{2}{\|\mathbf{w}\|}$



- Hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$
- Extra scale constraint:

  $\min_{i=1,\ldots,n} |\mathbf{w}^T\mathbf{x}_i + b| = 1$
- This implies:

  $\mathbf{w}^T(\mathbf{x}_a - \mathbf{x}_b) = 2$

  $\rho = \|\mathbf{x}_a - \mathbf{x}_b\|_2 = 2/\|\mathbf{w}\|_2$

Linear SVM Mathematically

– Then we can formulate the ***quadratic optimization problem:***

Find $\mathbf{w}$ and b such that

$\rho = \frac{2}{\|\mathbf{w}\|}$ is **maximized**; and for all $\{(\mathbf{x_i}, y_i)\}$

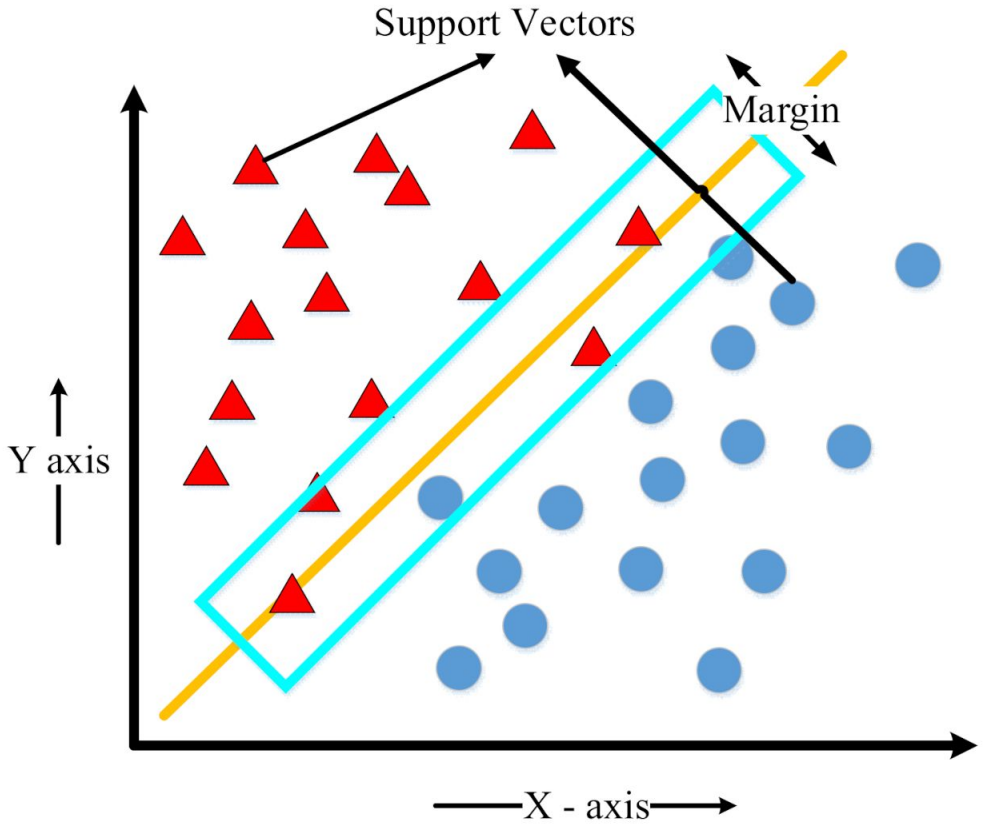
$\mathbf{w^T x_i} + b \geq 1$ if $y_i = 1$; $\mathbf{w^T x_i} + b \leq -1$ if $y_i = -1$

– A better formulation (min **||w||** = max 1/ **||w||**):

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w^T w}$ is **minimized**;

and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w^T x_i} + b) \geq 1$



Outline

- Vector and Hyperplane
- Main Ideas of SVM
- SVM Optimization Problem
- Non-Linear SVMs
- Introduction to Kernels

Solving the Optimization Problem

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q(\alpha)} = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^T \mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

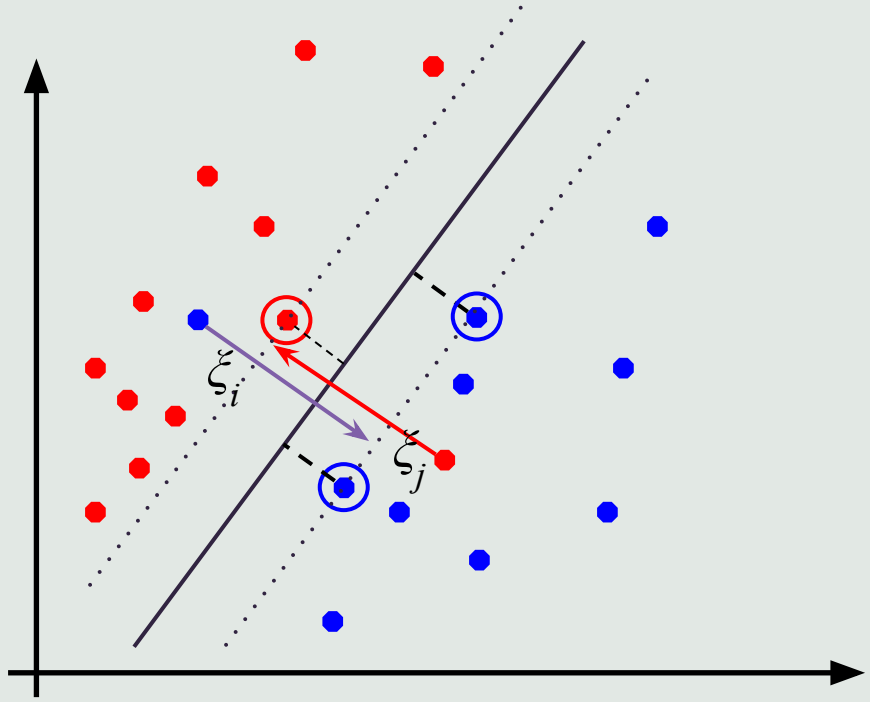
- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x_i}$
  - We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^T \mathbf{x_j}$ between all pairs of training points.

Soft Margin Classification

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane “far” from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathrm{b}) \geq 1$

- The new formulation incorporating slack variables:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter C can be viewed as a way to control overfitting
  - A regularization term

Soft Margin Classification Mathematically

- The dual problem for soft margin classification:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

- Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!
- Again, $\mathbf{x}_i$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i$
$b = y_k(1 - \xi_k) - \mathbf{w}^T\mathbf{x}_k$ where $k = \underset{k'}{\text{argmax}}\ \alpha_{k'}$

w is not needed explicitly for classification!

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$

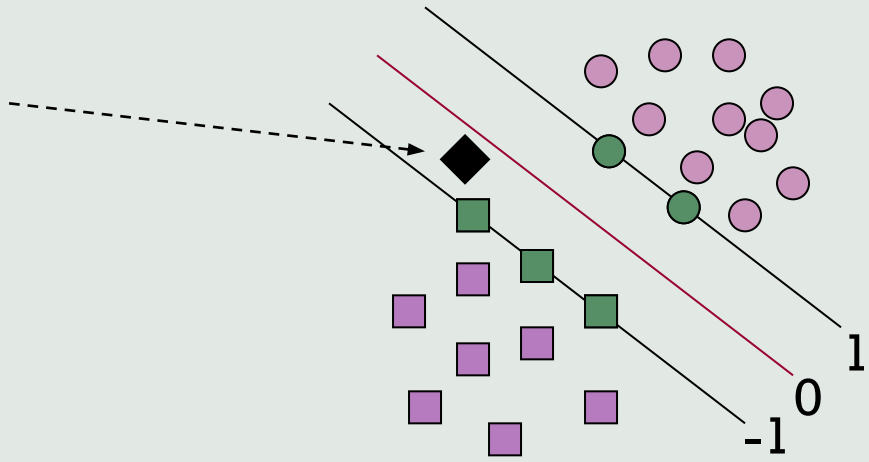
Classification with SVMs

- Given a new point **x**, we can score its projection onto the hyperplane normal:
  - I.e., compute score: $\mathbf{w}^T\mathbf{x} + b = \Sigma \alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$
    - Decide class based on whether < or > 0
  - Can set confidence threshold *t*.

Score > *t*: yes

Score < -*t*: no

Else: don't know

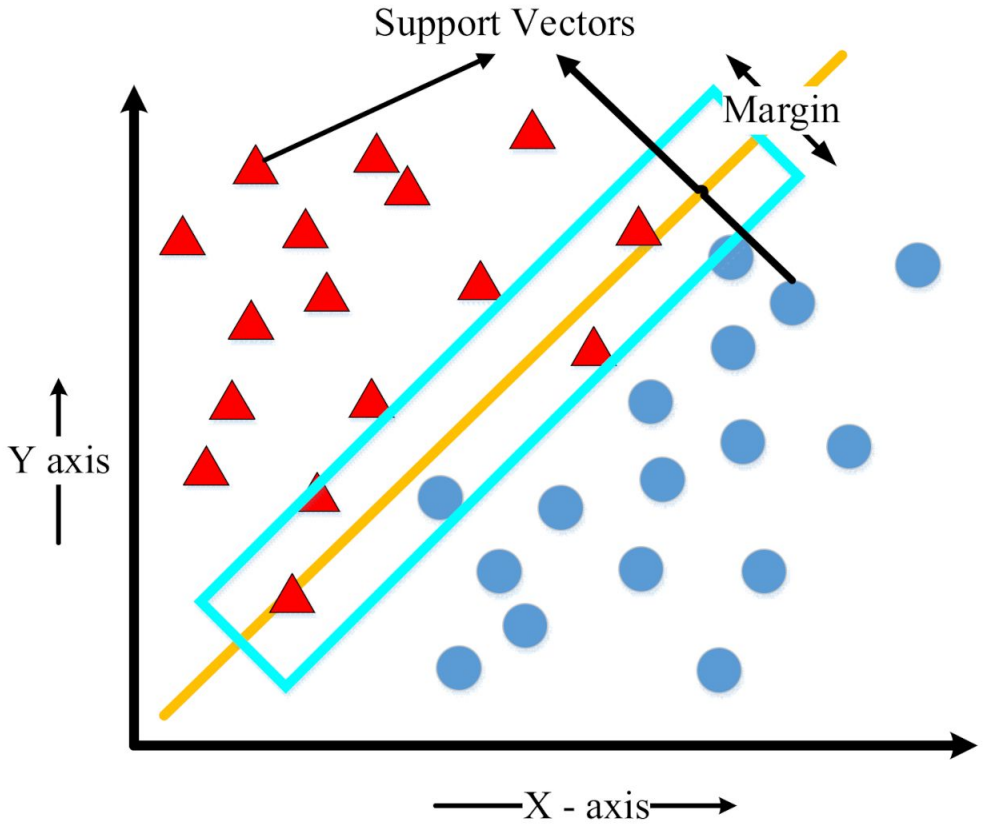


Linear SVMs: Summary

- The classifier is a *separating hyperplane.*
- The most "important" training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$



Outline

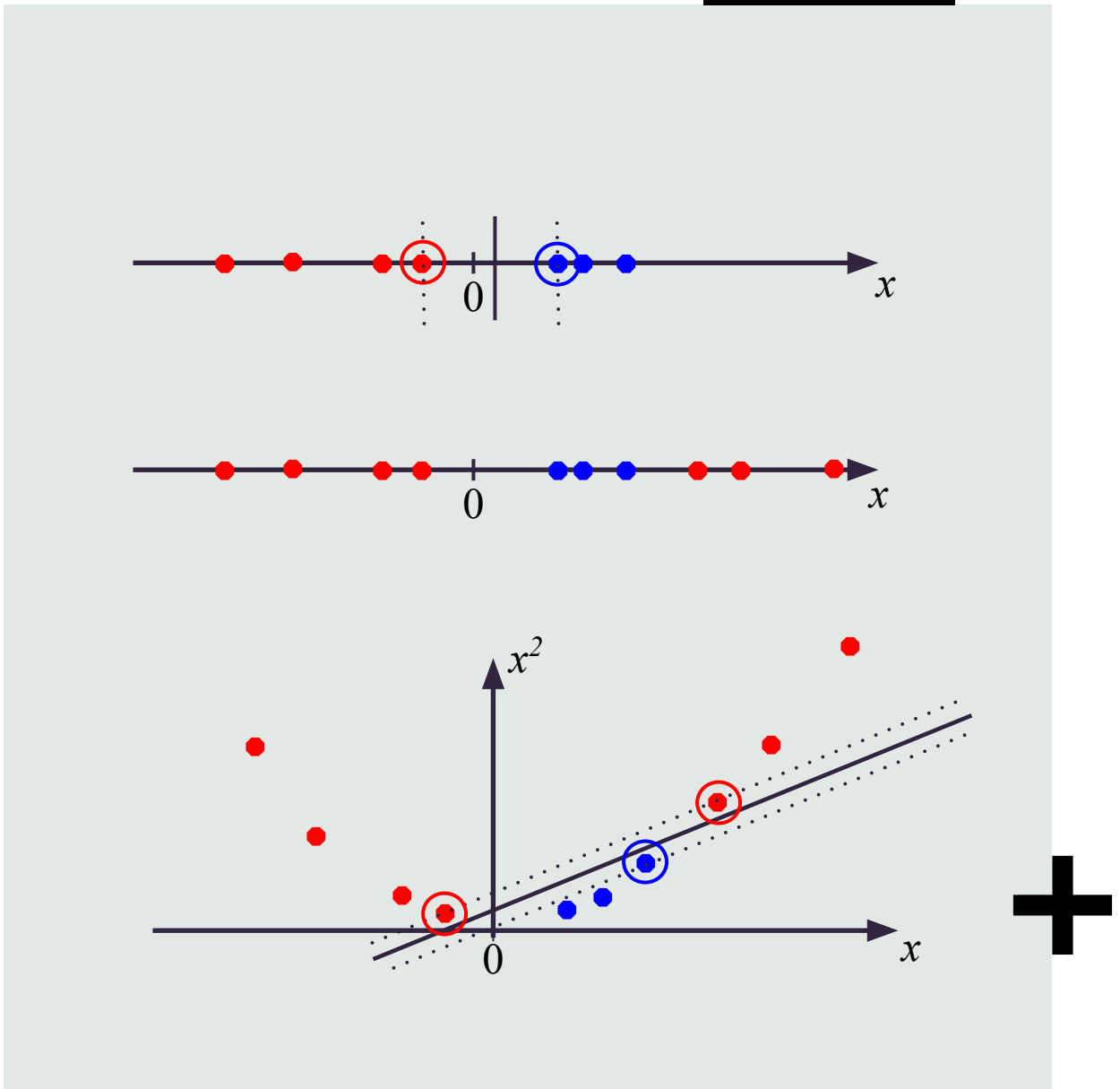
- Vector and Hyperplane
- Main Ideas of SVM
- SVM Optimization Problem
- Non-Linear SVMs
- Introduction to Kernels

Non-linear SVMs

- Datasets that are linearly separable (with some noise) work out great:

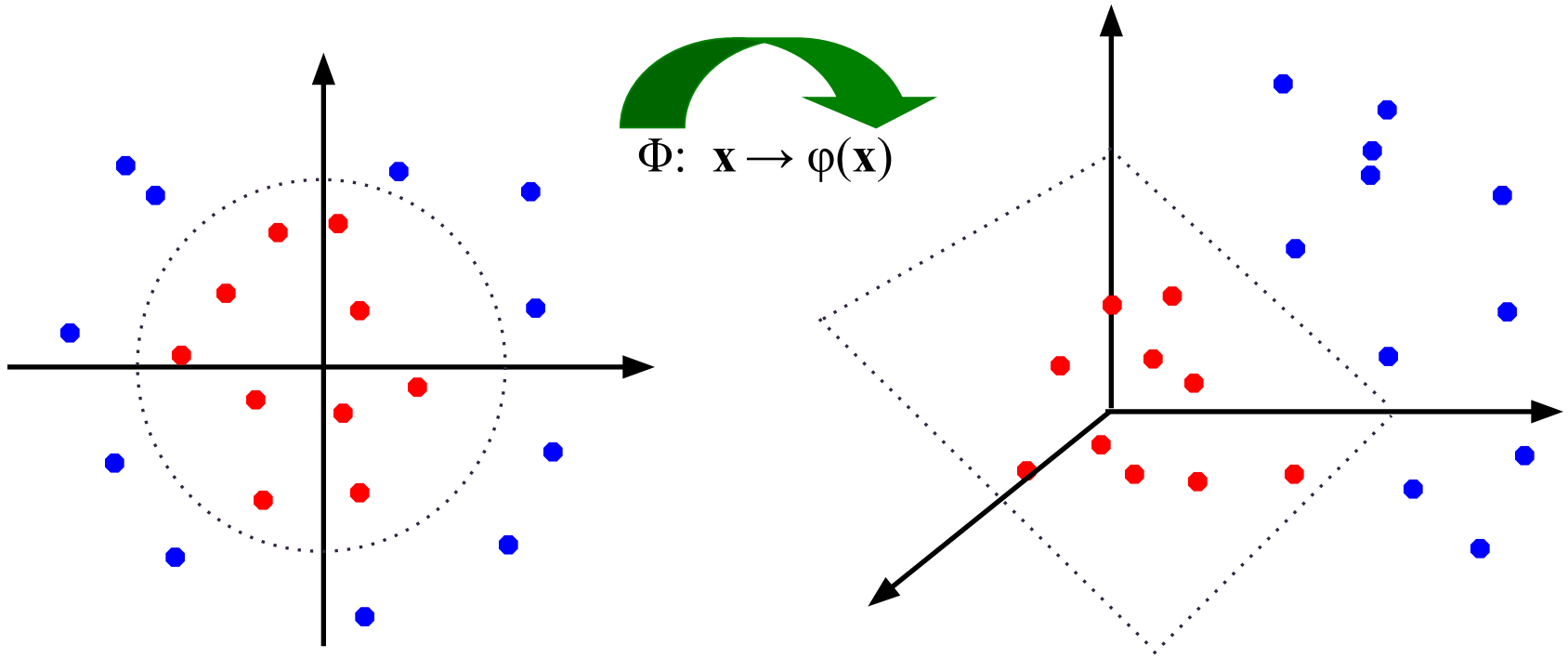
- But what are we going to do if the dataset is just too hard?

- How about … mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

– General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$:

  $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

  $= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$

  $= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$ where $\varphi(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

Kernels

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear
  - Polynomial $K(x,z) = (1+x^Tz)^d$
    - Gives feature conjunctions
  - Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$

- Haven't been very useful in text classification

Home Works

Scikit Learn

1.4. Support Vector Machines

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

The support vector machines in scikit-learn support both dense (`numpy.ndarray` and convertible to that by `numpy.asarray`) and sparse (any `scipy.sparse`) sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data. For optimal performance, use C-ordered `numpy.ndarray` (dense) or `scipy.sparse.csr_matrix` (sparse) with `dtype=float64`.

Kaggle

SVM Classifier Tutorial

Python · [Private Datasource]

Notebook Data Logs Comments (14)

Run
1334.1s

Version 4 of 4

Support Vector Machines Classifier Tutorial with Python

Hello friends,

Support Vector Machines (SVMs in short) are supervised machine learning algorithms that are used for classification and regression purposes. In this kernel, I build a Support Vector Machines classifier to classify a Pulsar star. I have used the **Predicting a Pulsar Star** dataset for this project.

So, let's get started.

As always, I hope you find this kernel useful and your UPVOTES would be highly appreciated.

Summary

- Support vector machines (SVM)
  - Choose hyperplane based on support vectors
    - Support vector = “critical” point close to decision boundary
  - (Degree-1) SVMs are linear classifiers.
  - Kernels: powerful and elegant way to define similarity metric
  - Perhaps best performing text classifier
    - But there are other methods that perform about as well as SVM, such as regularized logistic regression (Zhang & Oles 2001)
  - Partly popular due to availability of good software
    - SVMlight is accurate and fast – and free (for research)
    - Now lots of good software: libsvm, TinySVM, ….
- Comparative evaluation of methods
- Real world: exploit domain specific structure!