Master of Technology

U2/6: Computational Intelligence I

Support Vector Machines

Dr TIAN Jing
Institute of Systems Science,
National University of Singapore
Email: tianjing@nus.edu.sg

SVM

Source: http://burrsettles.com/miscellany
http://yann.lecun.com/ex/fun/index.html
https://www.csie.ntu.edu.tw/~cjlin/index.html

Module reference

- Prof. Andrew Zisserman (Oxford), Machine Learning lectures, http://www.robots.ox.ac.uk/~az/lectures/ml/
- M. Law, A Simple Introduction to Support Vector Machines, Michigan State University, available at https://www.cise.ufl.edu/class/cis4930sp11dtm/notes.html
- A. Géron, *Hands-on machine learning with scikit-learn and tensorFlow concepts, tools, and techniques to build*, O'Reilly Media, 2017. E-book available in NUS library, code available at https://github.com/ageron/handson-ml
- C. Cortes and V. N. Vapnik, "Support-vector networks“, *Machine Learning*, Vol. 20, No. 3, 1995, pp. 273-297.
- M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, “Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?” *Journal of Machine Learning Research*, Vol. 15, Oct. 2014, pp. 3133-3181.
- Prof. Patrick Winston (MIT), 50-minute lecture video, Support Vector Machines, https://www.youtube.com/watch?v=_PwhiWxHK8o

Preliminary

Scalar

24

Vector

$\begin{bmatrix} 2 & -8 & 7 \end{bmatrix}$ row

or column $\begin{bmatrix} 2 \\ -8 \\ 7 \end{bmatrix}$

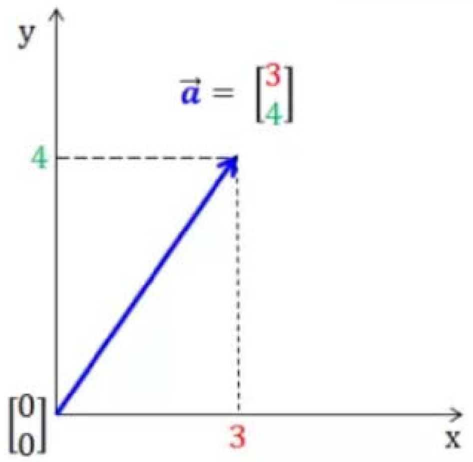
Matrix

$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}$

row(s) × column(s)

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} * y = \begin{bmatrix} a*y & b*y \\ c*y & d*y \end{bmatrix}$$

$$\begin{bmatrix} 4 & 8 \\ 3 & 7 \end{bmatrix} * 3 = \begin{bmatrix} 12 & 24 \\ 9 & 21 \end{bmatrix}$$



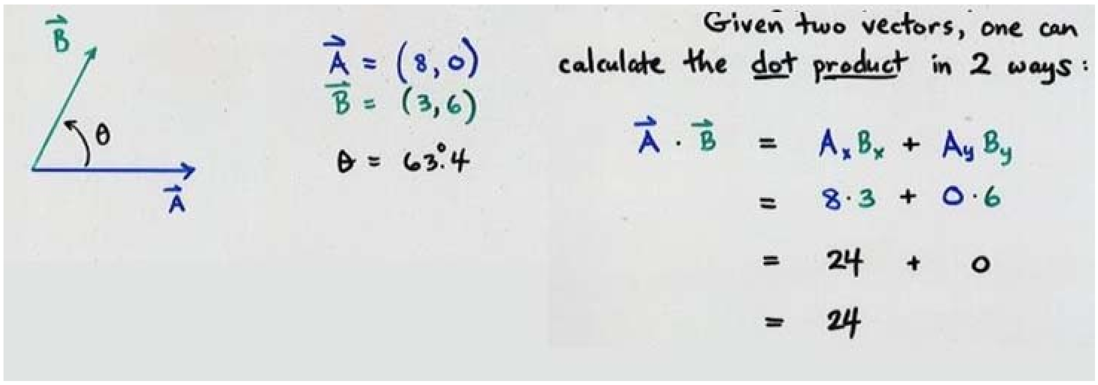
y
$\vec{a} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$
4
$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$
3
x

Preliminary

For $x, y \in \mathbf{R}^n$, the ***dot product*** of x and y, denoted $x \cdot y$, is defined by

$$x \cdot y = x_1 y_1 + \cdots + x_n y_n,$$

where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$.



Given two vectors, one can calculate the dot product in 2 ways:

$$\vec{A} \cdot \vec{B} = A_x B_x + A_y B_y = 8 \cdot 3 + 0 \cdot 6 = 24 + 0 = 24$$

Source: http://spiff.rit.edu/classes/phys311.old/lectures/dot/dot.html

Outline

- Understand SVM without mathematics
- Hands-on SVM programming
- Advanced theory of SVM

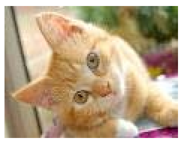
Classification

- Input: the description of a situation
- Output: a class label. It could represent a decision, a prediction, an action, etc.

Speech classification

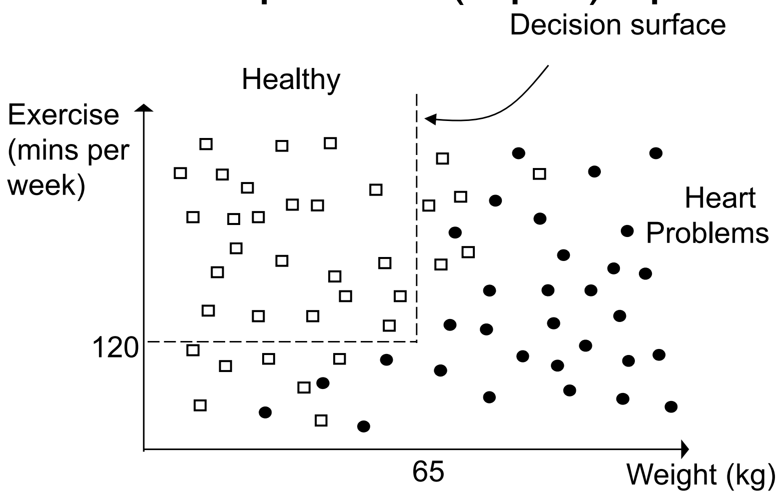
$f($ $) =$ "How are you"

Image classification

$f($  $) =$ "Cat"

Classification

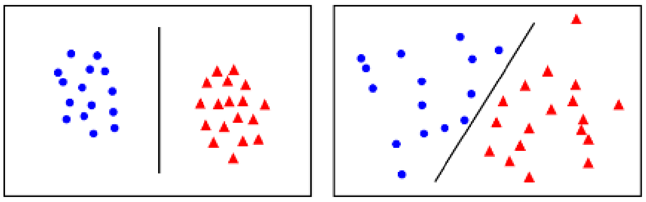
- Rule-based system
  - » **IF** (Weight < 65kg) **AND** (Exercise > 120mins) **THEN** healthy
- Decision surfaces in pattern (input) space.



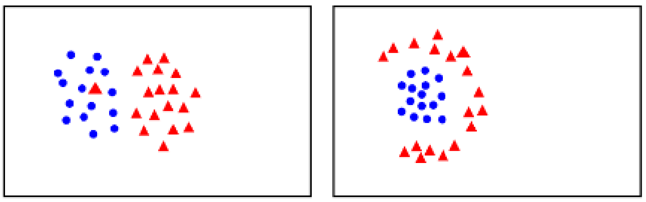
A special classification: Linear separability

- When a linear hyperplane exists to place the instances of one class on one side and those of the other class on the other side.

Linearly separable



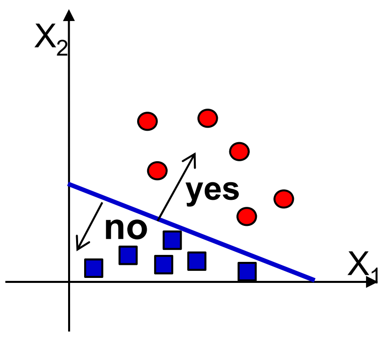
not linearly separable

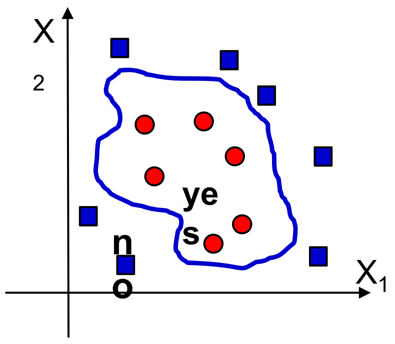


Linear classifier

- Learning for binary classification is formulated as: Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots l$, with $\mathbf{x}_i \in \mathbf{R}^n$ and $y_i \in \{-1, 1\}$, to learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i)\begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

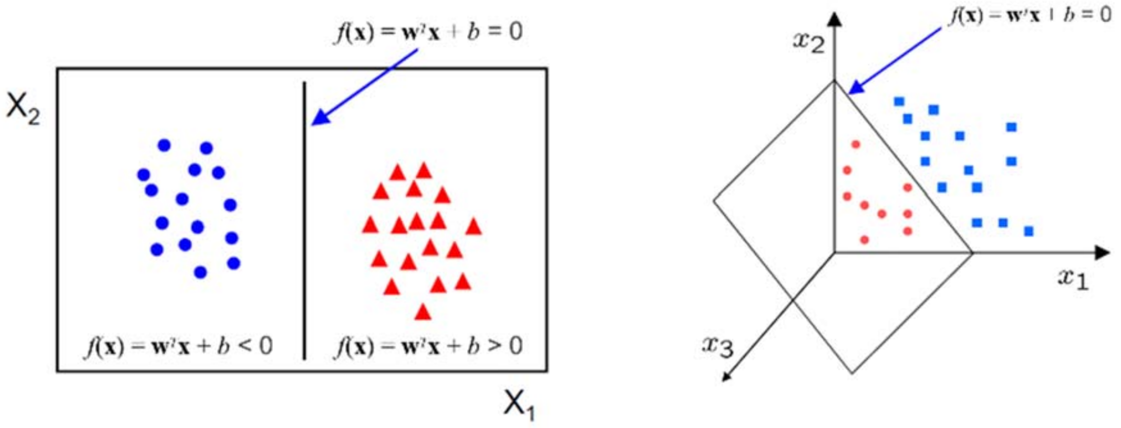




$y_i \cdot f(\mathbf{x}_i) \geq 0$ for a correct classification

Linear classifier

- A linear classifier has the form $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$



$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$
X_2
X_1
$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b < 0$
$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b > 0$
$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$
x_2
x_1
x_3

  - » In 2D space, the classifier is a line, $\mathbf{w}$ is weight vector, and b the bias.
  - » In 3D space, the classifier is a plane, and in *n*D space, it is a hyperplance.

Linear classifier

- Consider linearly separable data (l samples)
  - » Training samples: $\{(\mathbf{x}_i, y_i)\}$, $i = 1 \ldots l$

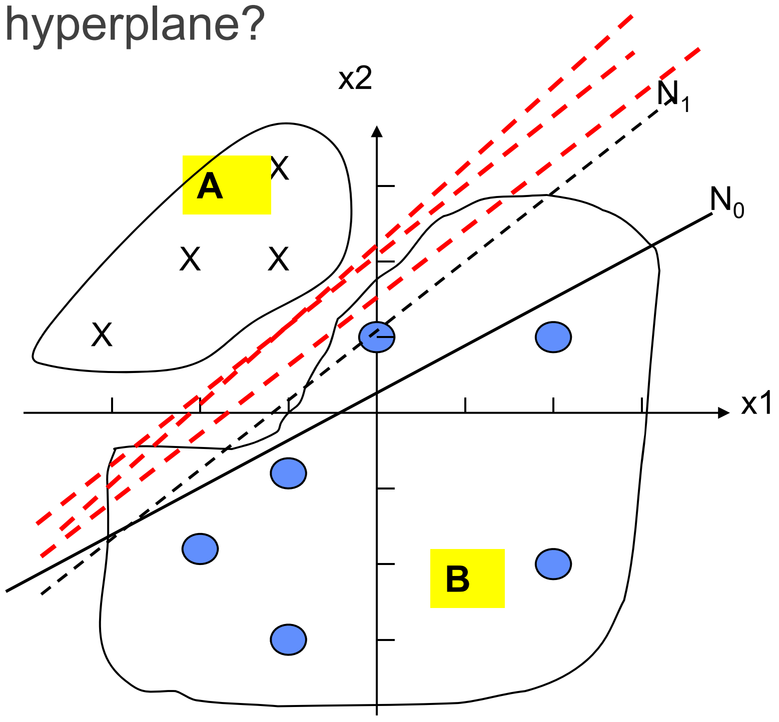
    $\mathbf{x}_i$: the input pattern for the *i*-th example

    $y_i \in \{-1,1\}$: the corresponding desired output
  - » The classifier for the separation is a hyperplane

i.e.
$$\boxed{\begin{array}{ll} \mathbf{w}^T\mathbf{x} + b = 0 & \\ \mathbf{w}^T\mathbf{x} + b \geq 0 & \text{for } y_i = 1 \\ \mathbf{w}^T\mathbf{x} + b < 0 & \text{for } y_i = -1 \end{array}}$$

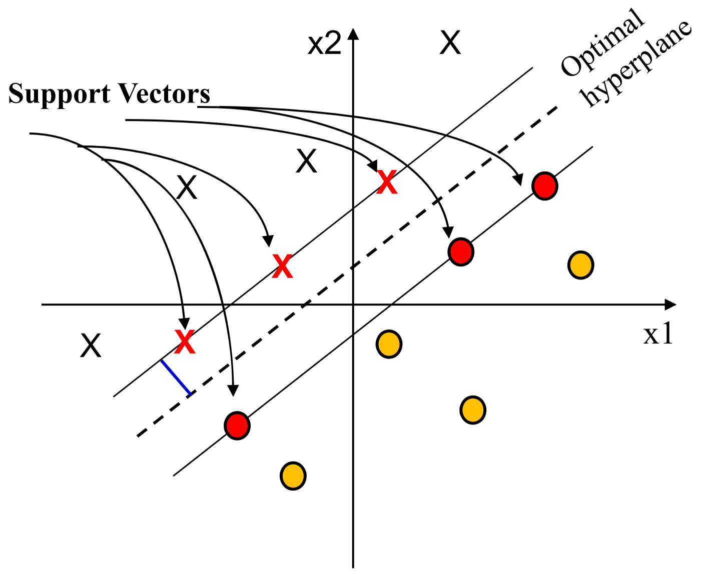
Linear classifier

Which hyperplane?



Linear classifier

- Optimal hyperplane is the particular hyperplane with the margin of separation maximized.

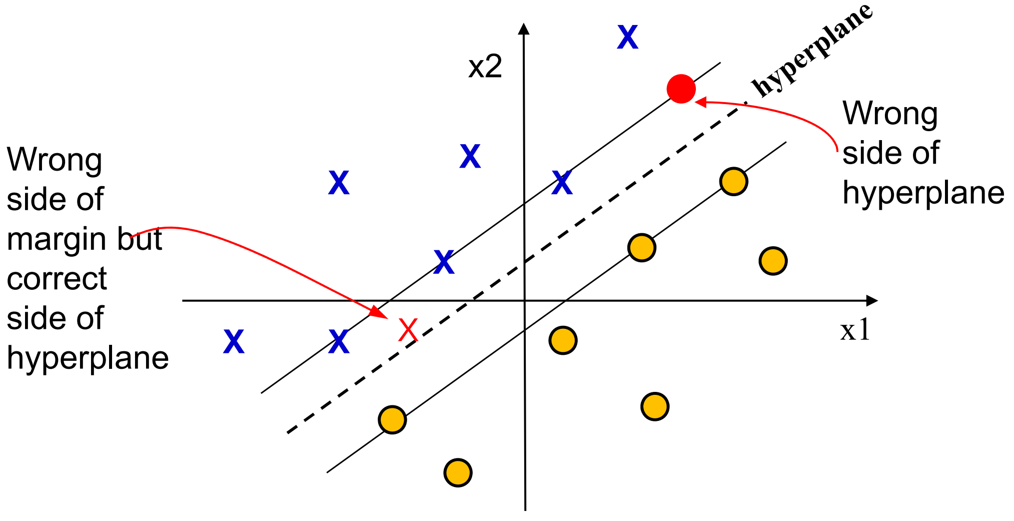


Next step

- Allowing a few errors in classification
  - » Soft margin
- Move forward to nonlinear classifier
  - » Kernel function
- Converting SVM to a form we can solve
  - » Learning SVM as an optimization

SVM: Soft margin

- Possible cases in support vector classifier
  - ✓ correct side of margin
  - X incorrect side of margin but correct side of hyperplane
  - ● incorrect side of hyperplane



SVM: Soft margin

- To find an optimal hyperplane that minimizes the probability of misclassification, averaged over the training set, that is we need to <u>minimize</u> $\sum_i \xi_i$, ξ_i can be computed by

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

  ≈ ξ_i are “slack variables” in optimization

  » Note that ξ_i=0 if there is no error for $\mathbf{x}_i$

New joint cost function: Maximum margin and misclassification error

Non-linear classifier: Idea

Mapping into feature space

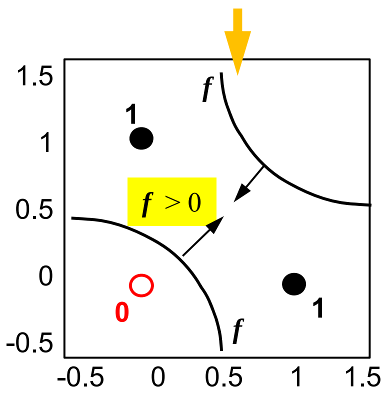
$$z_1 = x_1, \quad z_2 = x_2, \quad z_3 = x_1 x_2$$

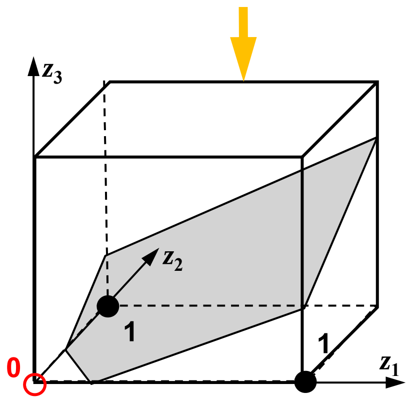
Nonlinear function

$$f(\mathbf{x}) = x_1 + x_2 - 2x_1x_2 - 1/3$$

Linear function in feature space

$$f(\mathbf{z}) = z_1 + z_2 - 2z_3 - 1/3$$





Linearly separable now !

Non-linear classifier: Idea

- To solve classification problem with non-linearly separable patterns
  - » Mapping
    - ♦ input space → feature space
  - » Problem to solve
    - ♦ non-linearly separable in input space

      → linearly separable in feature space
- Key idea
  - » Through an appropriate mapping, a hard problem can be made more solvable

Non-linear SVM classifier

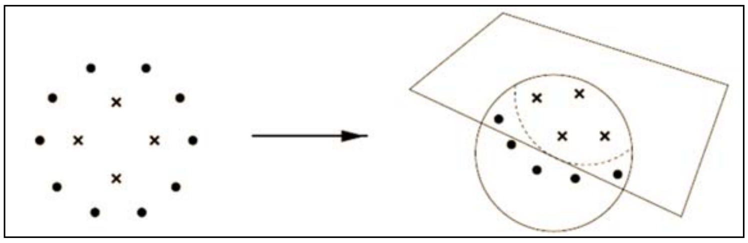
- First map data into a richer space including nonlinear features

$$\Phi : \ \mathbf{x} \ \alpha \ \ \phi(\mathbf{x})$$

- Then construct a hyperplane from the feature space

$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b$$

- The separating hyperplane as a linear function of vector drawn from the feature space rather than the original input space



Non-linear SVM classifier

- Example
  - » Map the original 2-dimensional input space to a 3-dimensional feature space

$$\begin{tikzcd} x = (x_1, x_2) \arrow[r, "\Phi(x)"'] \arrow[d] & (x_1, x_2, x_1x_2) \arrow[d] \\ \textit{Input space } \boldsymbol{X} & \textit{Feature space } \boldsymbol{F} \end{tikzcd}$$

  - » The original non-linearly separable problem becomes linearly separable in the feature space

$$\begin{tikzcd} (x_1, x_2, x_1x_2) \arrow[r, "\text{decision function}"', "y = w\cdot\phi(x) + b"' {yshift=-12pt}] \arrow[d] & y \in Y \arrow[d] \\ \textit{Feature space } \boldsymbol{F} & \textit{Target (output) space } Y \end{tikzcd}$$

Learning for classification

- The target of learning is to achieve a minimized error of classification with decision surface
- Using **dual representation** we can rewrite

$$d(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

Provided in sample data

Parameters associated with input vectors in sample data, $\alpha_i \neq 0$ for support vectors

$$d(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle + b$$

All the information the learning algorithm needs is the **inner products** between data points in the feature space, where $\mathbf{x}$, $\mathbf{x}_i$ $(i = 1, \ldots, l)$ $\in \mathbf{X}$, the input space

Kernel function

- A function that performs this direct computation of inner product is known as a kernel function, denoted by

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

- The kernel function is equivalent to the distance between **x** and **x′** measured in the higher dimensional feature space transformed by Φ

Kernel function: Example

- The inner product in the feature space can be evaluated as

$$\Phi : \quad \mathbf{x} = (x_1, x_2) \alpha \;\; \phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in F = \mathbf{R}^3$$

$$\begin{aligned}\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (x_1'^2, x_2'^2, \sqrt{2}x_1'x_2') \rangle \\ &= x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1x_2x_1'x_2' \\ &= (x_1x_1' + x_2x_2')^2 = \langle \mathbf{x}, \mathbf{x}' \rangle^2\end{aligned}$$

where $\mathbf{x}, \mathbf{x}' \in \mathbf{X}$

» Hence, the function $K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^2$ is a kernel function in input space, with F its corresponding feature space

Kernel function: Example

Input space $\vec{x_i} = (x_{i1}, x_{i2})$ Feature space $\vec{X_i} = (x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2})$

Inner product $\vec{X_i} \cdot \vec{X_j} = X_{i1}X_{j1} + X_{i2}X_{j2} + X_{i3}X_{j3}$

Without kernel trick

To get the new first dimension: 1 multiplication
Second dimension: 1 multiplication
Third dimension: 2 multiplications
In all, 1+1+2 = 4 multiplications.
Multiplications: 8 (for the projections) + 3 (in the dot product) = 11 multiplications
Additions: 2 (in the dot product)
Total: 11 + 2 = 13 operations.

With kernel trick

$$\begin{aligned} K(\vec{x_i}, \vec{x_j}) &= (\vec{x_i} \cdot \vec{x_j})^2 \\ &= (x_{i1}x_{j1} + x_{i2}x_{j2})^2 \\ &= x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} \\ &= (x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2}) \cdot (x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2}) \end{aligned}$$

Source: https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93

Multiplications: 2 (for the dot product in the original space) + 1 (for squaring the result) = 3 multiplications
Additions: 1 (for the dot product in the original space)
Total: 3 + 1 = 4 operations.

Examples of Kernel functions

- Polynomial kernel with degree *d*

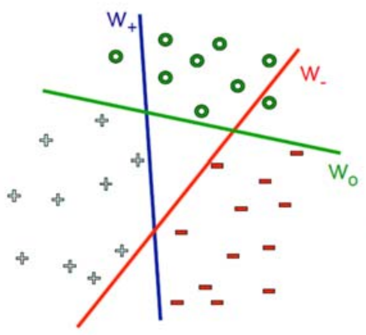
$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

Multi-class SVM classifier

- One vs. others
  - » Training: Learn an SVM for each vs. the others
  - » Testing: Apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value



- Learn 3 classifiers:
  - – - vs. {o,+}, weights w_-
  - – + vs. {o,-}, weights w_+
  - – o vs. {+,-}, weights w_o
- Predict label using:

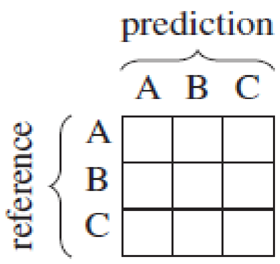
$$\hat{y} \leftarrow \arg\max_k \; w_k \cdot x + b_k$$

- One vs. one
  - » Training: Learn an SVM for each pair of classes
  - » Testing: Major voting from each learned SVM

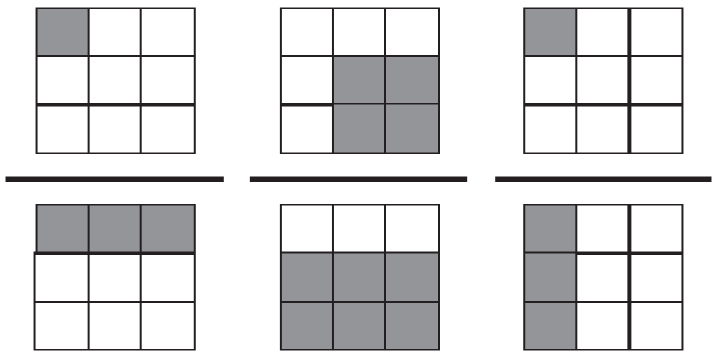
Performance evaluation (1)

For class A

We need to repeat for each individual class.



Sensitivity Specificity Precision



Confusion matrix		Prediction		
		A	B	C
Reference	A	5	2	0
	B	1	4	1
	C	0	2	10

Sensitivity

$$\boxed{\frac{5}{5+2+0} = 0.71}$$

Precision

$$\boxed{\frac{5}{5+1+0} = 0.83}$$

Specificity

$$\boxed{\frac{4+1+2+10}{1+4+1+0+2+10} = 0.94}$$

Performance evaluation (2)

- True positives (TP): The data that is correctly classified by a model as positive instance of the concept being modelled.
- False positives (FP): The data that is classified as positive instance by the model, but in fact are known not to be.
- True negatives (TN): The data correctly classified by the model as not being instances of the concept.
- False negatives (FN): The data that is classified as not being instances, but are in fact know to be.

- **Classifier Accuracy**
  - **Accuracy = (TP + TN)/All**
- **Sensitivity**: True Positive recognition rate
  - **Sensitivity = TP/P**
- **Specificity**: True Negative recognition rate
  - **Specificity = TN/N**

- **Precision**: How much of data that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** How much of positive data did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$

SVM for regression

- The general regression learning problem is set as
  - » The learning machine is given the training data with l observations

$$\boldsymbol{D} = \{(\mathbf{x}_i, y_i)\}_l \quad \mathbf{x} \in \mathbf{R}^n \text{ — n-dimensional vectors}$$
$$y \in \mathrm{R} \text{ — continuous values}$$

from which it attempts to learn the input-output relationship

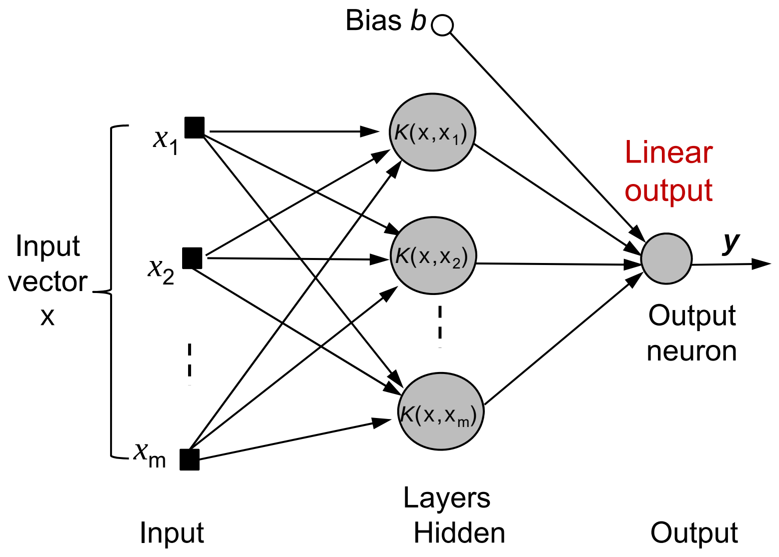
- SVM considers approximating function of the form

$$f(\mathbf{x}, \mathbf{w}) = \sum_i w_i \phi_i(\mathbf{x})$$

where $\phi_i(\mathbf{x})$ are same as in nonlinear classification

SVM and neural network

- General architecture
  - » Input layer
  - » Hidden layer of inner-product kernels (fully connected with the input layer)
  - » Output neuron for a linear function of hidden neurons' response



SVM and neural network

- Neural network is a computational model that mimics the pattern of the human mind
- SVM first map input data into a high dimensional feature space defined by kernel function, and find the optimum hyperplane that separates the training data by the maximum margin
  - » We can think of SVM as a linear algorithm in a high dimensional space (transformed from input space through non-linear mapping)

SVM and neural network

- They differ by the learning method used
  - » NNs typically use BP (back propagation) or gradient descent algorithm
  - » SVMs model the learning problem as optimization, then solve it as QP (quadratic programming) or LP (linear programming) problem

Outline

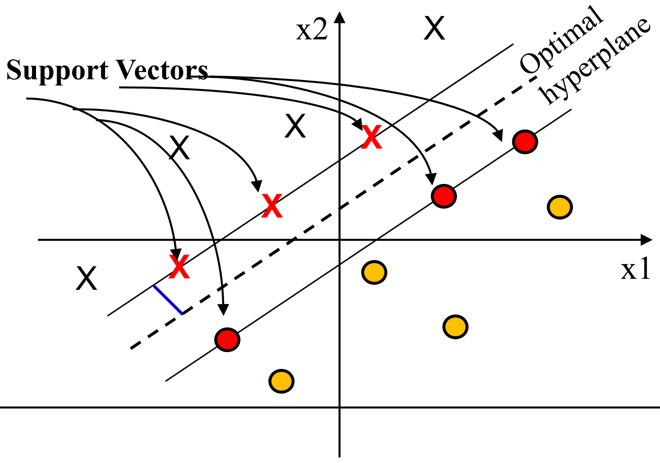
- Understand SVM without mathematics
- Hands-on SVM programming
- Advanced theory of SVM

SVM: Revisit

- Support vector machine is a method of obtaining the optimal boundary of two sets in a vector space independently on the probabilistic distributions of training vectors in the sets.
  - » maximal margin classifier
    - ♦ finds optimal hyperplane for linearly separable patterns (hard margin)
  - » support vector classifier
    - ♦ introduces soft margin to allow possible misclassification
  - » support vector machine
    - ♦ classifies patterns that are not linearly separable by transforming original data through kernel function

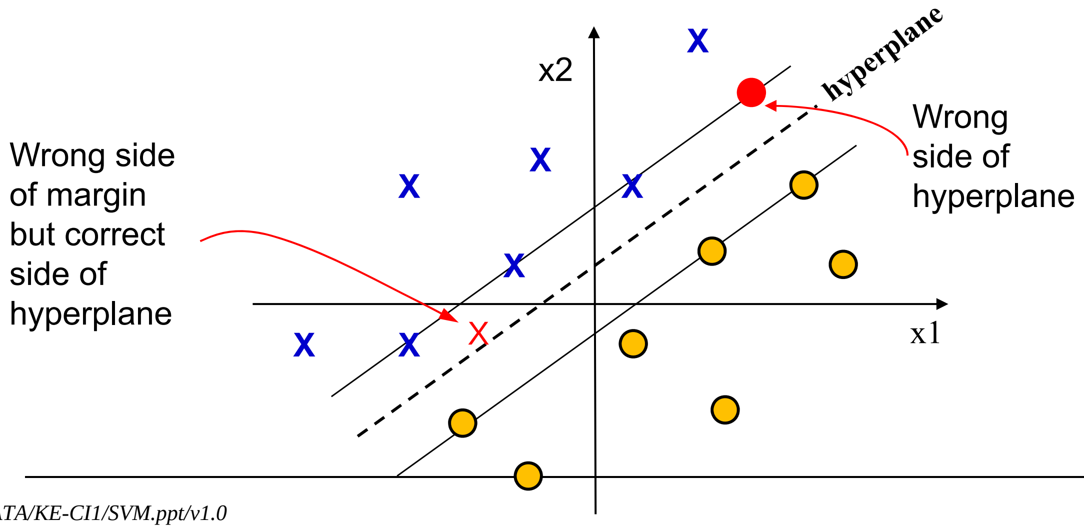
SVM: Maximal margin

- Maximal margin classifier for linearly separable case is based on maximal margin hyperplane. The maximal margin hyperplane depends directly only on the support vectors, but not on the other observations



SVM: Soft margin

- Possible cases in support vector classifier
  - ✓ correct side of margin
  - X incorrect side of margin but correct side of hyperplane
  - ● incorrect side of hyperplane



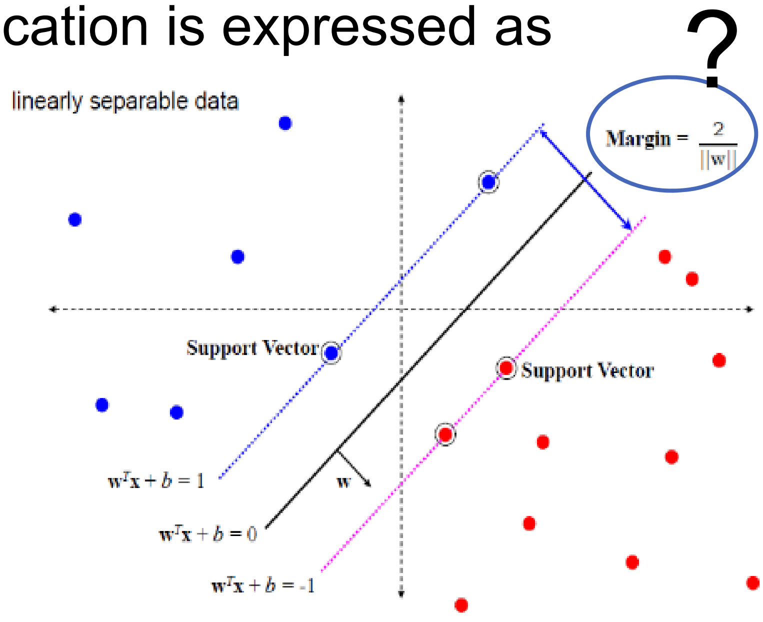
SVM: Kernel

- An extension of the support vector classifier that results from enlarging the feature space in a specific way, using kernels
- A kernel is a function that quantifies the similarity of two observations
  - » Kernel trick
- Types of kernels
  - » Polynomial kernel
  - » Gaussian (radial-basis) kernel

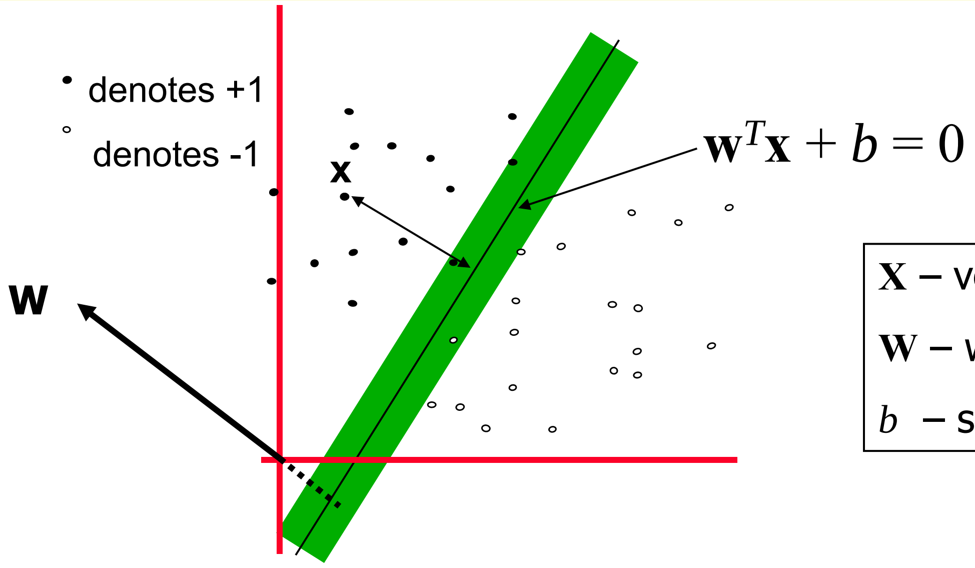
Classifier hyperplane

- A hyperplane for classification is expressed as
  $$\mathbf{w}^T\mathbf{x}+b=0$$
- The distance between a training data vector $\mathbf{x}_i$ and the boundary is expressed as

$$\frac{\left|\mathbf{w}^T\mathbf{x}_i+b\right|}{\left\|\mathbf{w}\right\|}$$


?
Margin = $\frac{2}{||\mathbf{w}||}$
linearly separable data
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x}+b=1$
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+b=-1$
$\mathbf{w}$

Classifier hyperplane



$\mathbf{X}$ – vector
$\mathbf{W}$ – weight vector
b – scale value

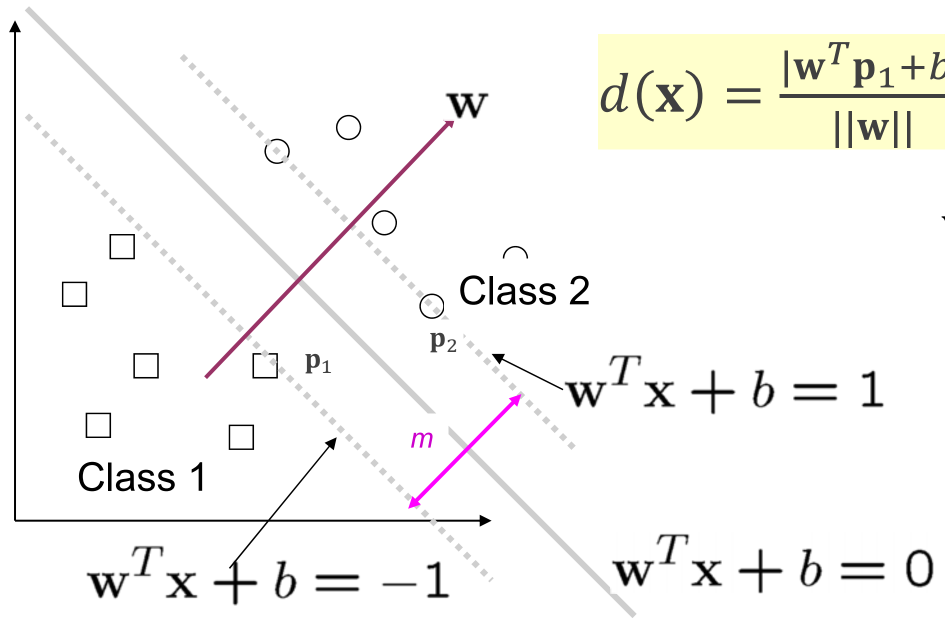
What is the distance expression for a point $\mathbf{x}$ to a line $\mathbf{w}^T\mathbf{x}+b=0$?

$$d(\mathbf{x})=\frac{|\mathbf{w}^T\mathbf{x}+b|}{||\mathbf{w}||}$$

Source: http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Classifier hyperplane

- To maximize the margin, which is the summation of distance between the classifier hyperplane to two support vectors $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively.



w
Class 2
$\mathbf{p}_2$
$\mathbf{p}_1$
m
Class 1
$\mathbf{w}^T\mathbf{x}+b=1$
$\mathbf{w}^T\mathbf{x}+b=-1$
$\mathbf{w}^T\mathbf{x}+b=0$

$$d(\mathbf{x}) = \frac{|\mathbf{w}^T\mathbf{p}_1+b|}{||\mathbf{w}||}+\frac{|\mathbf{w}^T\mathbf{p}_2+b|}{||\mathbf{w}||}=\frac{2}{||\mathbf{w}||}$$

$$\mathbf{w}^T\mathbf{p}_1 + b = -1$$

$$\mathbf{w}^T\mathbf{p}_2 + b = 1$$

ATA/KE-CI1/SVM.ppt/v1.0

Learning SVM as optimization

$$\text{maximize } \frac{2}{||\mathbf{w}||} \Rightarrow \boxed{\begin{array}{lll} \text{minimize} & \frac{1}{2}\mathbf{w}^T\mathbf{w} & \\ \text{subject to} & y_i(\mathbf{w}^T\mathbf{x}_i+b)\geq 1 & \forall i \end{array}}$$

where $\mathbf{w}$ satisfy

$$\mathbf{w}^T\mathbf{x}_i+b\geq+1 \quad \text{for } y_i=+1$$

$$\mathbf{w}^T\mathbf{x}_i+b\leq-1 \quad \text{for } y_i=-1$$

Justification

We are minimizing $\frac{1}{2}\mathbf{w}^T\cdot\mathbf{w}$, which is equal to $\frac{1}{2}\|\mathbf{w}\|^2$, rather than minimizing $\|\mathbf{w}\|$. This is because it will give the same result (since the values of $\mathbf{w}$ and b that minimize a value also minimize half of its square), but $\frac{1}{2}\|\mathbf{w}\|^2$ has a nice and simple derivative

See Page 158, A. Géron, *Hands-on machine learning with scikit-learn and tensorFlow concepts, tools, and techniques to build*, O'Reilly Media, 2017.

ATA/KE-CI1/SVM.ppt/v1.0

Learning SVM as optimization

Constraint equations

$$L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \, [y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1] \qquad \text{Eq. (1)}$$

Objective equations

where α_i, $i = 1, \ldots, l$, $\alpha_i \geq 0$ are Lagrange multipliers, or indeterminate coefficient

Learning SVM as optimization

- Search for an optimal solution is achieved by
  - » either in a primal space (which is the space of parameters $\mathbf{w}$ and b), by minimizing L_P
  - » or in a dual space (which is the space of Lagrange multipliers α_i), by maximizing L_D
- From Eq. (1), if $\mathbf{w}$ and b take the optimal value, the partial derivates are zero

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i \ , \qquad \text{Eq. (2)}$$

$$\frac{\partial L_P}{\partial b} = -\sum_i \alpha_i y_i \qquad \text{Eq. (3)}$$

Learning SVM as optimization

- Setting the derivates of Eq. (2) to zero, we get

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \ , \qquad \text{Eq. (4)}$$

$$\sum_i \alpha_i y_i = 0 \qquad \text{Eq. (5)}$$

- Substituting Eq. (4) and Eq. (5) to the primal Lagrangian in Eq. (1), with necessary rewriting (see next slide), we obtain the dual Lagrangian

$$L_D(\boldsymbol{\alpha}) = \sum_i^l \alpha_i - \frac{1}{2}\sum_{i,j}^l \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \qquad \text{Eq. (6)}$$

to be maximized with respect to non-negative α_i, $i = 1, \ldots, l$

Appendix: Derivation of Eq. (6)

According to Eq. (1) in PPT, we have

$$L_p(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right] = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{w}^T\mathbf{x}_i - \sum_{i=1}^{l} \alpha_i y_i b + \sum_{i=1}^{l} \alpha_i$$

Substitute $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$ (see Eq. (4) in PPT) and $\sum_{i=1}^{l} \alpha_i y_i = 0$ (see Eq. (5) in PPT) into above equation, we have

$$\begin{aligned} L_p(\mathbf{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2}\mathbf{w}^T \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i - \mathbf{w}^T \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i + \sum_{i=1}^{l} \alpha_i = -\frac{1}{2}\mathbf{w}^T \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i + \sum_{i=1}^{l} \alpha_i \\ &= -\frac{1}{2}\left(\sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i\right)^T \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i + \sum_{i=1}^{l} \alpha_i = -\frac{1}{2}\sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i^T \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i + \sum_{i=1}^{l} \alpha_i \\ &= -\frac{1}{2}\sum_{i,j=1}^{l} \alpha_i y_i \mathbf{x}_i^T \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^{l} \alpha_i = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \mathbf{x}_i^T \mathbf{x}_j \end{aligned}$$

This is Eq. (6) in PPT.

Learning SVM as optimization

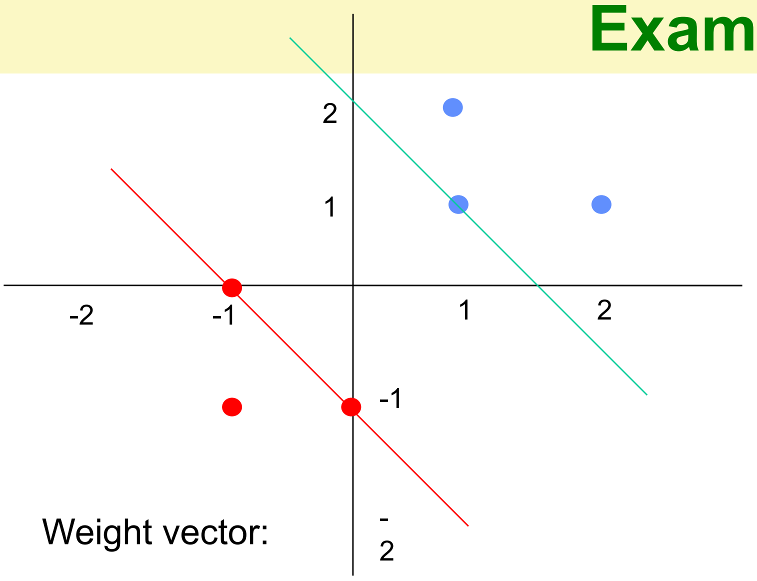
- SVM for linear case, decision hyperplane is given by

l training samples $\{(x_i, y_i)\}_l$

$$d(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{l} y_i \alpha_i \underbrace{\langle \mathbf{x}, \mathbf{x}_i \rangle}_{\text{Inner product}} + b \qquad \boxed{\text{Eq. (7)}}$$

Substitute $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$ $\boxed{\text{Eq. (4)}}$

Example



Input to SVM optimizer:

x_1	x_2	y
1	1	1
1	2	1
2	1	1
−1	0	−1
0	−1	−1
−1	−1	−1

Weight vector:

$$\mathbf{w} = \sum_{k \in \{\text{training examples}\}} \alpha_k \mathbf{x}_k$$

$$= -.208\,(-1,0) + .416\,(1,1) - .208\,(0,-1)$$

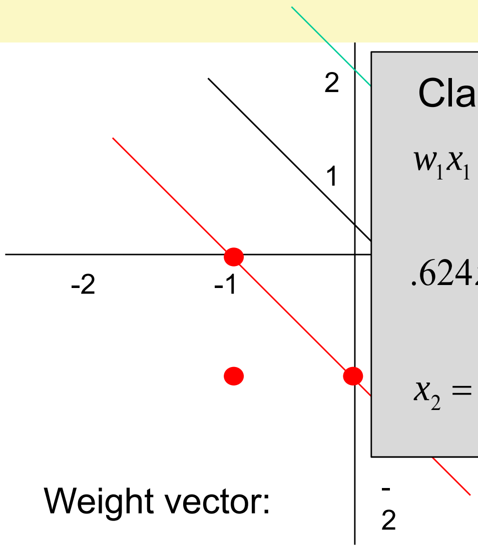
$$= (.624,.624)$$

Output from SVM optimizer:

Support vectors	
(−1, 0)	−.208
(1, 1)	.416
(0, −1)	−.208

b = −.376

Example



Classifier hyperplane

$$w_1x_1 + w_2x_2 + b = 0$$

$$.624x_1 + .624x_2 - .376 = 0$$

$$x_2 = -x_1 + .6$$

Input to SVM optimizer:

x_1	x_2	y
1	1	1
1	2	1
2	1	1
−1	0	−1
0	−1	−1
−1	−1	−1

Weight vector:

$$\mathbf{w} = \sum_{k \in \{\text{training examples}\}} \alpha_k \mathbf{x}_k$$
$$= -.208\,(-1,0) + .416\,(1,1) - .208\,(0,-1)$$
$$= (.624, .624)$$

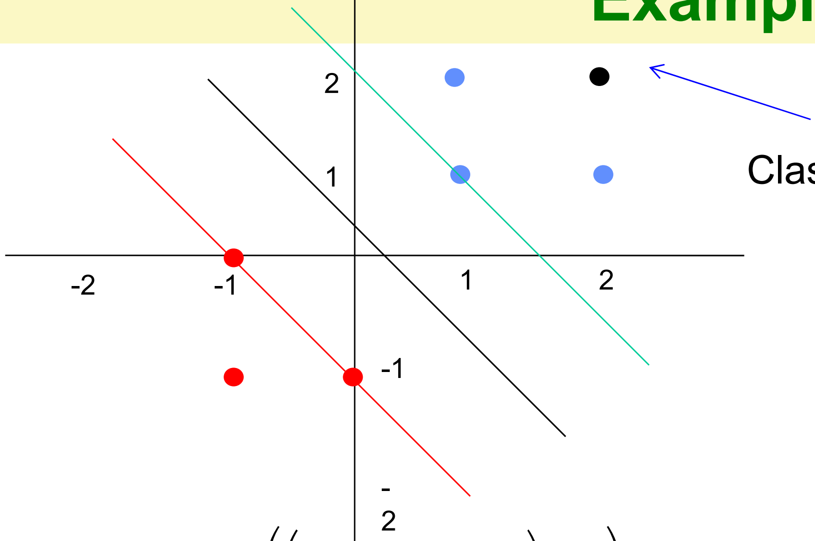
Output from SVM optimizer:

Support vectors

(−1, 0)	−.208
(1, 1)	.416
(0, −1)	−.208

b = −.376

Example



Classifying a new point (2,2)

$$h((2,2)) = \operatorname{sgn}\left(\left(\sum_{k=1}^{m} \alpha_k (\mathbf{x}_k \cdot \mathbf{x})\right) + b\right), \quad \text{where } \operatorname{sgn}(z) = \begin{cases} 1 \text{ if } z > 0 \\ -1 \text{ if } z \le 0 \end{cases}$$

$$= \operatorname{sgn}\left(-.208[(-1,0)\cdot(2,2)] + .416[(1,1)\cdot(2,2)] - .208[(0,-1)\cdot(2,2)] - .376\right)$$

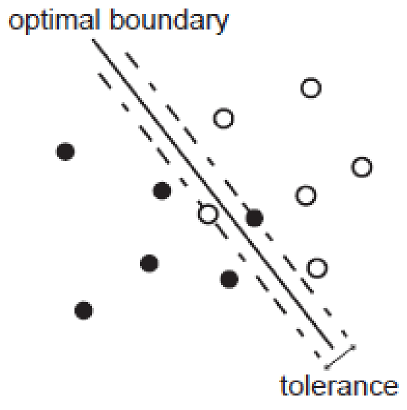
$$= \operatorname{sgn}(.416 + 1.664 + .416 - .376) = +1$$

SVM: Soft margin solution

- To classify data sets that are not linearly separable, the SVM within the linear framework is extended by introducing soft margin
  - » Replace the restriction

    $$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

    where ξ_i , called slack variables, are positive variables that indicate tolerance of misclassification.



$$\boxed{\begin{array}{lll} \text{minimize} & \frac{1}{2}\mathbf{w}^T\mathbf{w} & \\ \text{subject to} & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 & \forall i \end{array}}$$

SVM: Soft margin solution

- There are optimization functions proposed for the case with soft margin, such as

$$\text{minimize} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

  - » C is a penalty parameter
    - ♦ small $C \Rightarrow$ wider margin (more tolerant)
      - many SVs will be on the margin or violate the margin
    - ♦ large $C \Rightarrow$ narrow margin
      - there will be few SVs on the margin or violating the margin
    - ♦ $C \rightarrow \infty$ enforces all constraints $\Rightarrow$ hard margin

SVM: Nonlinear classifier

- The decision hyperplane given in Eq. (7) is extended
  - » the vectors from input space $\mathbf{x}$, $\mathbf{x}_i$ are replaced by their images in the transformed feature space: $\phi(\mathbf{x})$ and $\phi(\mathbf{x}_i)$

$$d(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i \underbrace{\langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle}_{} + b$$

l training samples $\{(\mathrm{x}_i, y_i)\}_l$

Kernel function

$$\boxed{\alpha_i} \geq 0$$

Only those data points nearest to the hyperplane have non-zero coefficient

Support vectors

SVM: Nonlinear classifier

- Instead of explicitly computing the transformation $\phi(\mathbf{x})$, we realize a nonlinear decision boundary in the original input space through a kernel function

$$\sum_i \alpha_i y_i \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

SVM: Nonlinear classifier

- Given a set of l training samples

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$$

with the corresponding set of input vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ and a kernel function $K(. \, , .)$ to evaluate the inner products in a feature space with feature map Φ,

  » We can form a symmetric ***l***-by-***l*** **kernel matrix**

$$\mathbf{K} = \{\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle\}_{i,j=1}^{l} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^{l}$$

Examples of Kernel functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2 / (2\sigma^2))$$

Example

- Objective: Classification for 1-D data
- Suppose we have 5 training data points
  - » $x_1=1$, $x_2=2$, $x_3=4$, $x_4=5$, $x_5=6$, with 1, 2, 6 as class A and 4, 5 as class B $\Rightarrow$ $y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$
- We use the polynomial kernel $K(a,b) = (ab+1)^2$ and C is set to 100. We need to find α_i (i=1, …, 5) by

$$\text{max.} \quad \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Example

- After solving optimization problem, we get
  - ≈ $\alpha_1=0$, $\alpha_2=2.5$, $\alpha_3=0$, $\alpha_4=7.333$, $\alpha_5=4.833$
  - » The support vectors are $\{x_2=2, x_4=5, x_5=6\}$
- For a new point z, the discriminant function is

$$\begin{aligned} & f(z) \\ &= 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + 4.833(1)(6z+1)^2 + b \\ &= 0.6667z^2 - 5.333z + b \end{aligned}$$

- *b* is sovled by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2 and x_5 lie on the line $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = 1$ and x_4 lies on the line $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = -1$
- All three give b=9

$$\boxed{f(z) = 0.6667z^2 - 5.333z + 9}$$

SVM in practice

- Prepare the dataset
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
  - » You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Test data can be classified using the α_i and the support vectors

SVM

Thank you!

Dr TIAN Jing
Email: tianjing@nus.edu.sg