

Weka(2) & SVM

2012-11-01

Eun-Sol Kim

Data format for Weka (.ARFF)

```
@relation heart-disease-simplified

@attribute age numeric
@attribute sex { female, male }
@attribute chest_pain_type { typ_angina, asympt, non_anginal, atyp_angina }
@attribute cholesterol numeric
@attribute exercise_induced_angina { no, yes }
@attribute class { present, not_present }

@data
63,male,typ_angina,233,no,not_present
67,male,asympt,286,yes,present
67,male,asympt,229,yes,present
38,female,non_anginal,?,no,not_present
```

Header

Data
(CSV format)

Note: You can easily generate 'arff' file by adding a header to a usual CSV text file

How to Evaluate the Performance? (1/2)

- Usually, build a Confusion Matrix out of given data
- Evaluation Metrics
 - Accuracy (percent correct)
 - Precision
 - Recall
 - Many other metrics: F-measure, Kappa score, etc.
- For fair evaluation, the ‘cross-validation’ scheme is used

How to Evaluate the Performance? (2/2)

- Confusion Matrix

Prediction \ Real	Positive	Negative	
Positive	TP	FP	All with positive Test
Negative	FN	TN	All with Negative Test
	All with Disease	All without Disease	Everyone

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}$$

As **recall** ↑ **precision** ↓

conversely:

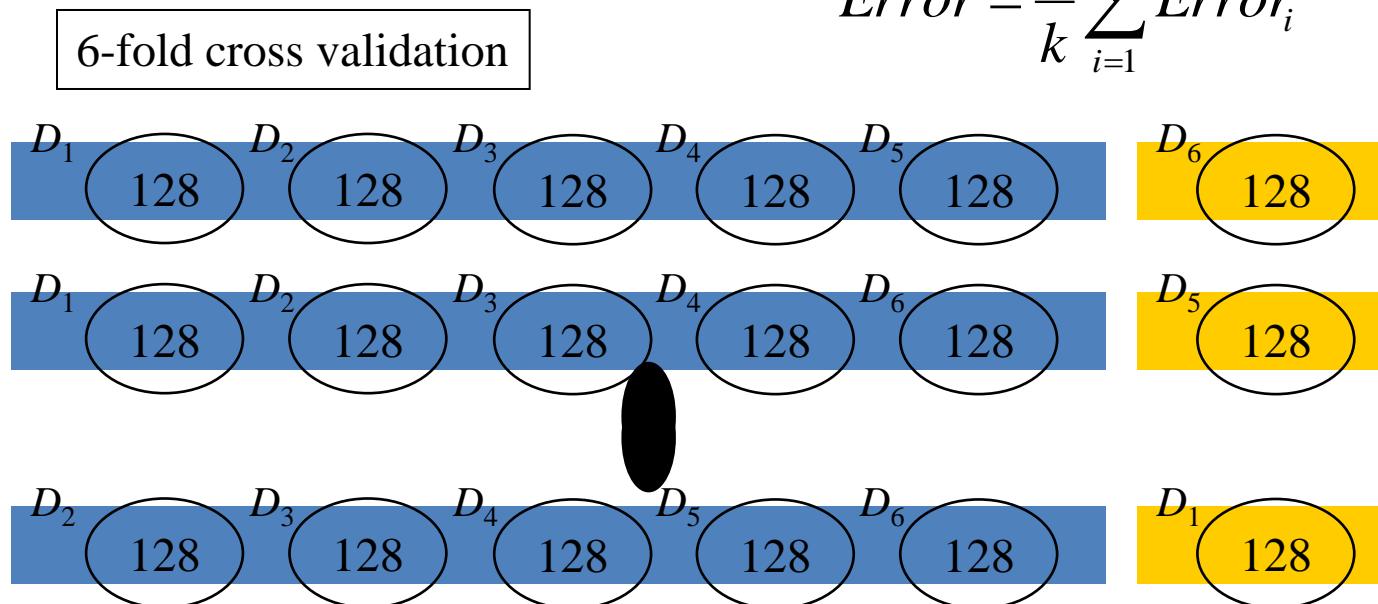
As **recall** ↓ **precision** ↑

Evaluation Method - Cross Validation

- *K*-fold Cross Validation

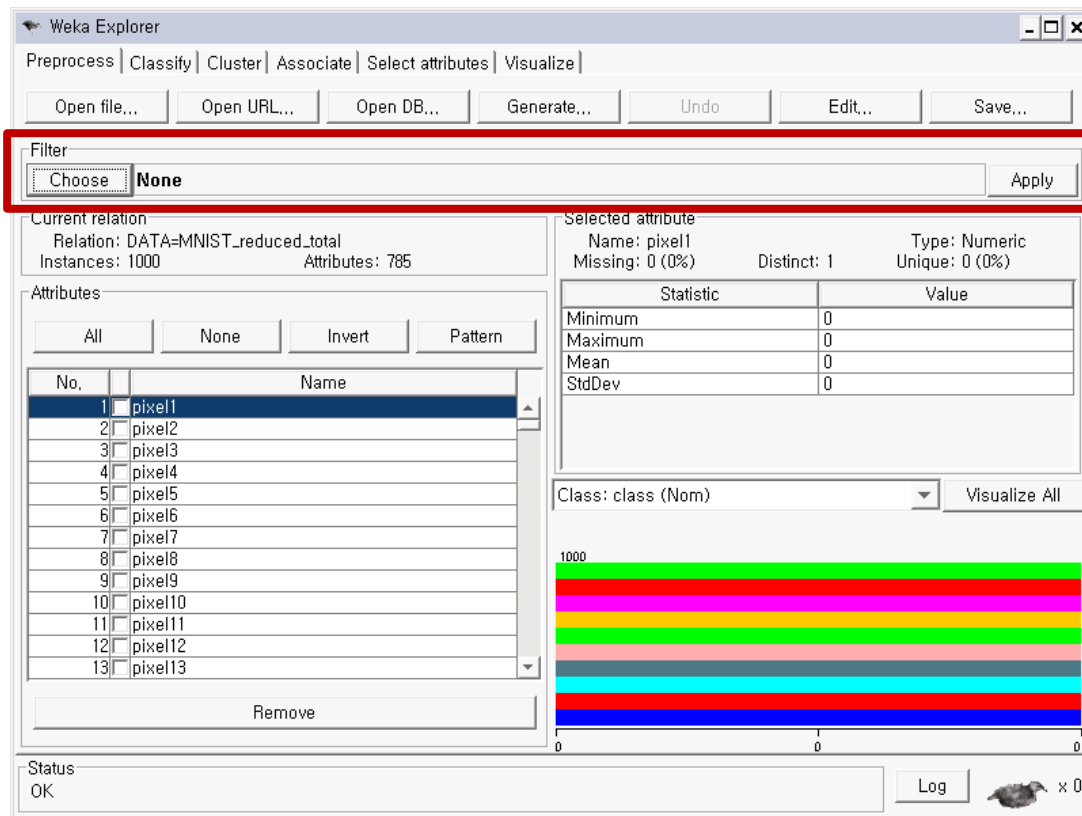
- The data set is randomly divided into *k* subsets.
- One of the *k* subsets is used as the ‘test set’ and the other *k*-1 subsets are put together to form a ‘training set’.

$$Error = \frac{1}{k} \sum_{i=1}^k Error_i$$



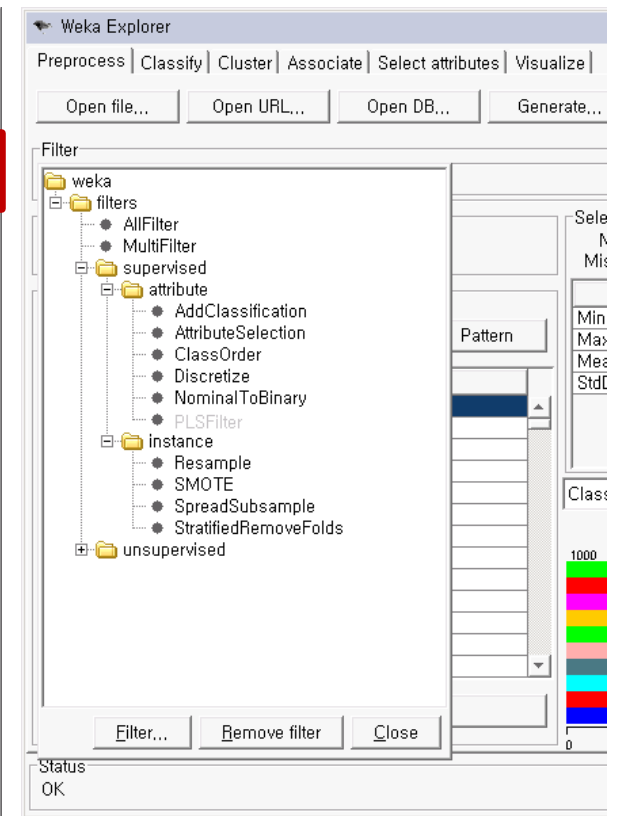
Data Manipulation with Filter in Weka

- Attribute
 - Selection, discretize
- Instance
 - Re-sampling, selecting specified folds



The screenshot shows the Weka Explorer interface with the Filter dialog box open. The dialog has a 'Choose' button selected and the filter name set to 'None'. Below the dialog, the 'Current relation' is 'DATA=MNIST_reduced_total' with 1000 instances and 785 attributes. The 'Attributes' list shows 13 pixel attributes. The 'Selected attribute' section shows 'pixel1' with a mean of 0. A small visualization at the bottom shows 1000 instances with various colored bars.

Statistic	Value
Minimum	0
Maximum	0
Mean	0
StdDev	0



The screenshot shows the Weka Explorer interface with the Filter dialog box open, displaying a tree view of available filters. The tree is expanded to show the 'supervised' category, which includes 'attribute' and 'instance' sub-categories. The 'attribute' category contains filters like 'AddClassification', 'AttributeSelection', 'ClassOrder', 'Discretize', and 'NominalToBinary'. The 'instance' category contains 'Resample', 'SMOTE', 'SpreadSubsample', and 'StratifiedRemoveFolds'. The 'unsupervised' category is also visible.

Using Experimenter in Weka

- Tool for 'Batch' experiments

Weka GUI Chooser

Application
Experimenter
KnowledgeFlow
Simple CLI

Weka Experiment Environment

Click 'New'

Setup | Run | Analyse

Experiment Configuration Mode: Simple

Open... Save... New

Results Destination
ARFF file | Filename: | Browse...

Experiment Type
Cross-validation
Number of folds: |
Classification | Regression

Iteration Control
Number of repetitions: |
Data sets first | Algorithms first

Datasets
Add new... Edit select... Delete select...
Use relative pat...

Algorithms
Add new... Edit selected... Delete selected

Weka Experiment Environment

Setup | Run | Analyse

Experiment Configuration Mode: Simple

Open...

Results Destination
ARFF file | Filename: | Browse...

Experiment Type
Cross-validation
Number of folds: 10
Classification | Regression

Iteration Control
Number of repetitions: 10
Data sets first | Algorithms first

Datasets
Add new... Edit select... Delete select...
Use relative pat...

E:\WPProgram Files\Weka-3-6\data\Wiris.arff

Algorithms
Add new... Edit selected... Delete selected
MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
MultilayerPerceptron -L 0.3 -M 0.2 -N 100 -V 0 -S 0 -E 20 -H a
MultilayerPerceptron -L 0.1 -M 0.2 -N 100 -V 0 -S 0 -E 20 -H a

Up Down Load options... Save options... Up Down

Notes

Weka Experiment Environment

Setup | Run | Analyse

Source
No source
File... Database... Experiment

Configure test
Testing with: Paired T-Tester (co...
Row: Select
Column: Select
Comparison field: |
Significance: 0.05
Sorting (asc.) by: |
Test base: Select
Displayed Columns: Select
Show std. deviations:
Output Format: Select

Test output

Perform test Save output

Weka Experiment Environment

Setup | Run | Analyse

Log
08:54:21: Started

• Select 'Run' tab and click 'Start'
• If it has finished successfully, click 'Analyse' tab and see the summary

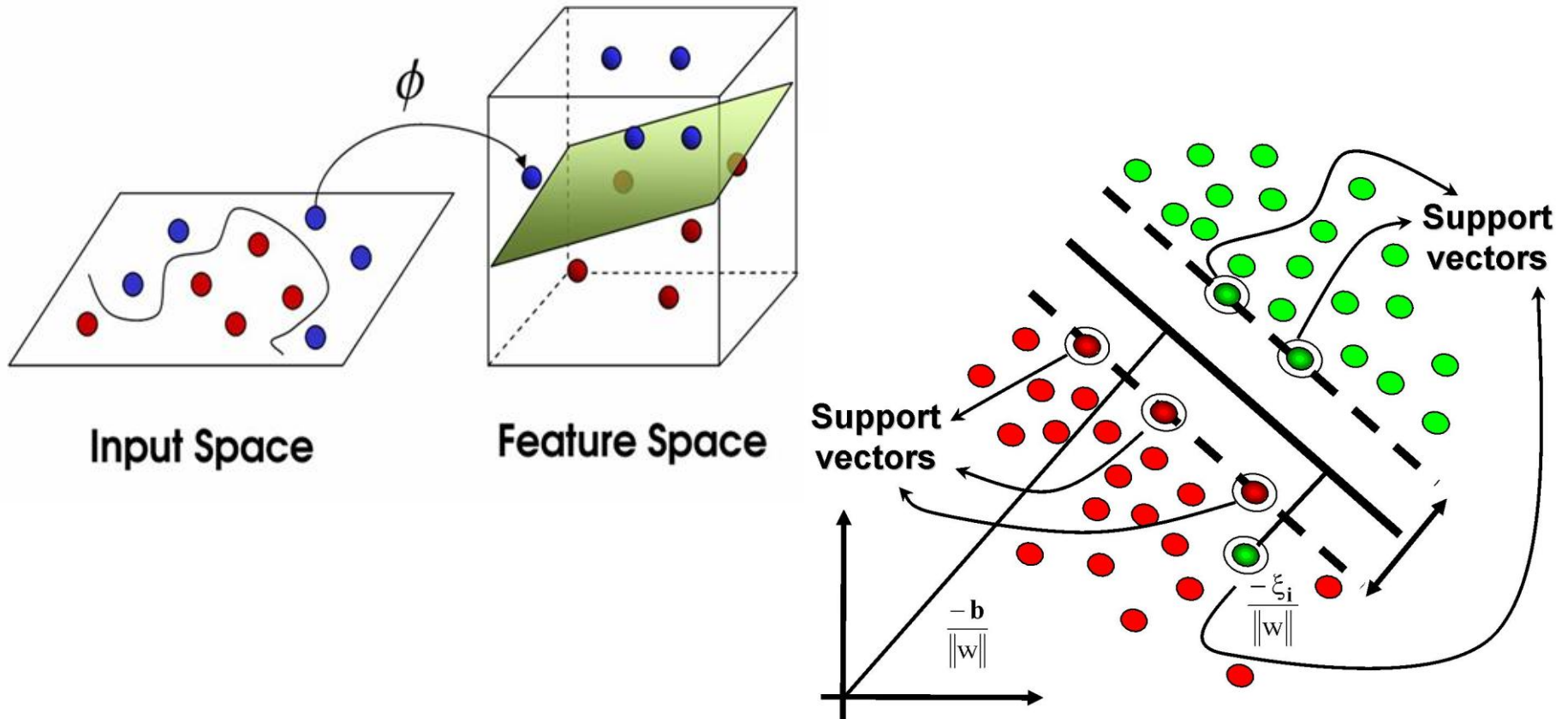
• Set experiment type/iteration control
• Set datasets / algorithms

(C) 2010, SNU Biointelligence Lab, <http://bi.snu.ac.kr/>

SVM

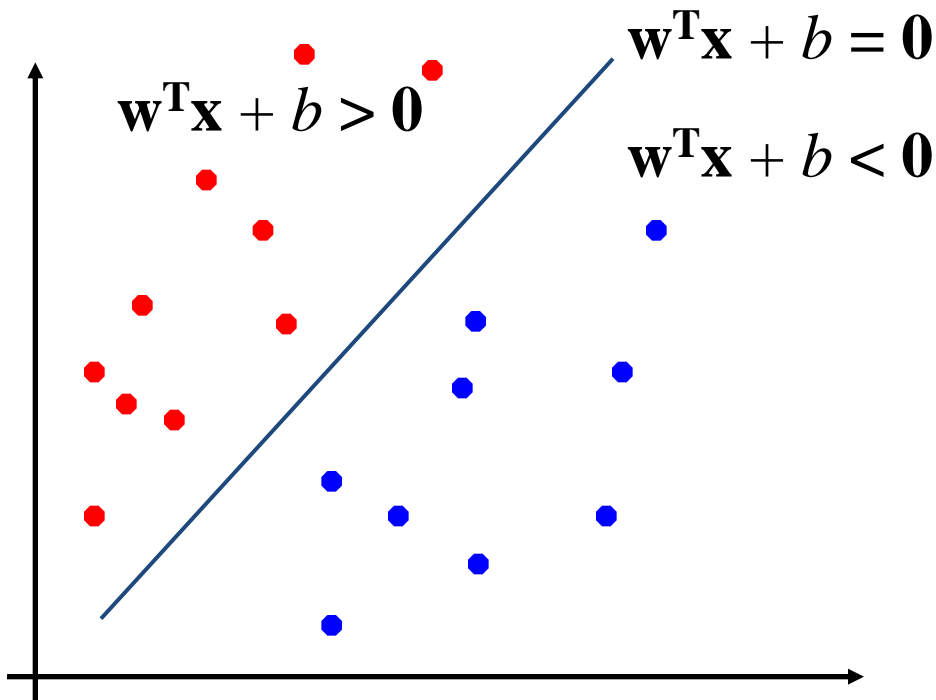
Support Vector Machines

- **SMO** (sequential minimal optimization) for training SVM
 - In Weka, classifiers-functions-SMO



Perceptron Revisited: Linear Separators

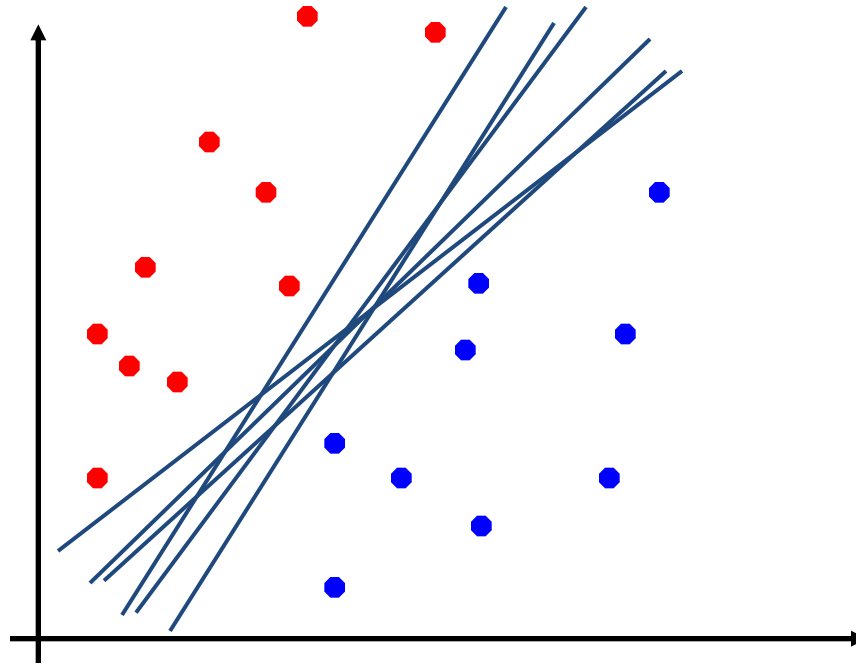
- Binary classification can be viewed as the task of separating classes in feature space:



$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

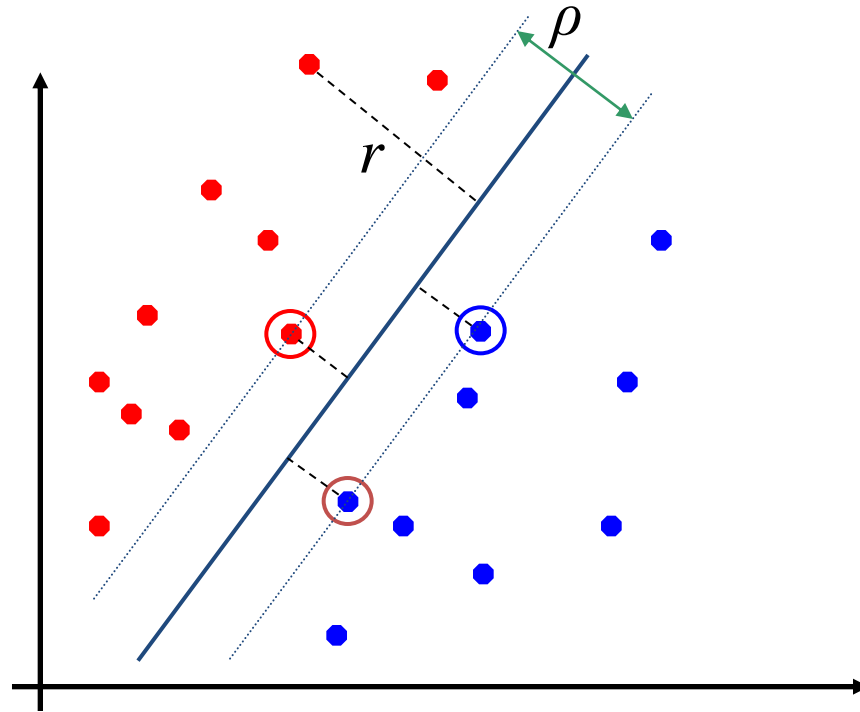
Linear Separators

- Which of the linear separators is optimal?



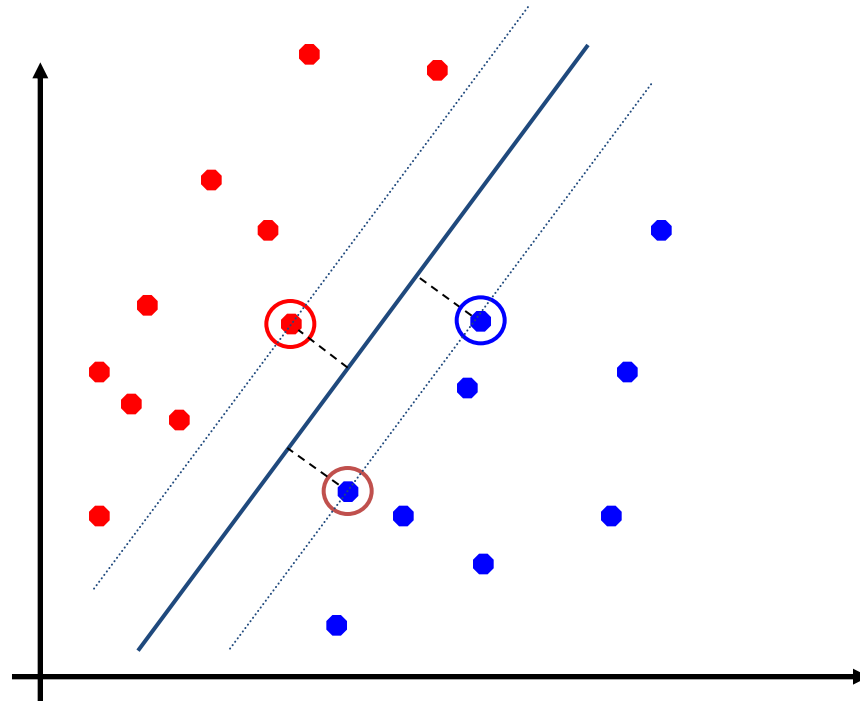
Classification Margin

- 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.



Linear SVM Mathematically

- Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ . Then for each training example (\mathbf{x}_i, y_i) :

$$\begin{aligned} \mathbf{w}^T \mathbf{x}_i + b &\leq -\rho/2 & \text{if } y_i = -1 \\ \mathbf{w}^T \mathbf{x}_i + b &\geq \rho/2 & \text{if } y_i = 1 \end{aligned} \quad \Leftrightarrow \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq \rho/2$$

- For every support vector \mathbf{x}_s the above inequality is an equality. After rescaling \mathbf{w} and b by $\rho/2$ in the equality, we obtain that distance between each \mathbf{x}_s and the hyperplane is $r = \frac{y_s(\mathbf{w}^T \mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$
- Then the margin can be expressed through (rescaled) \mathbf{w} and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Linear SVMs Mathematically (cont.)

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

Find \mathbf{w} and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ is maximized}$$

and for all $(\mathbf{x}_i, y_i), i=1..n$: $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

Which can be reformulated as:

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w} \text{ is minimized}$$

and for all $(\mathbf{x}_i, y_i), i=1..n$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

Solving the Optimization Problem

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \mathbf{w}^T \mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i), i=1..n$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every inequality constraint in the primal (original) problem:

Find $\alpha_1 \dots \alpha_n$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

(1) $\sum \alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- Given a solution $\alpha_1 \dots \alpha_n$ to the dual problem, solution to the primal is:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \quad b = y_k - \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_k \quad \text{for any } \alpha_k > 0$$

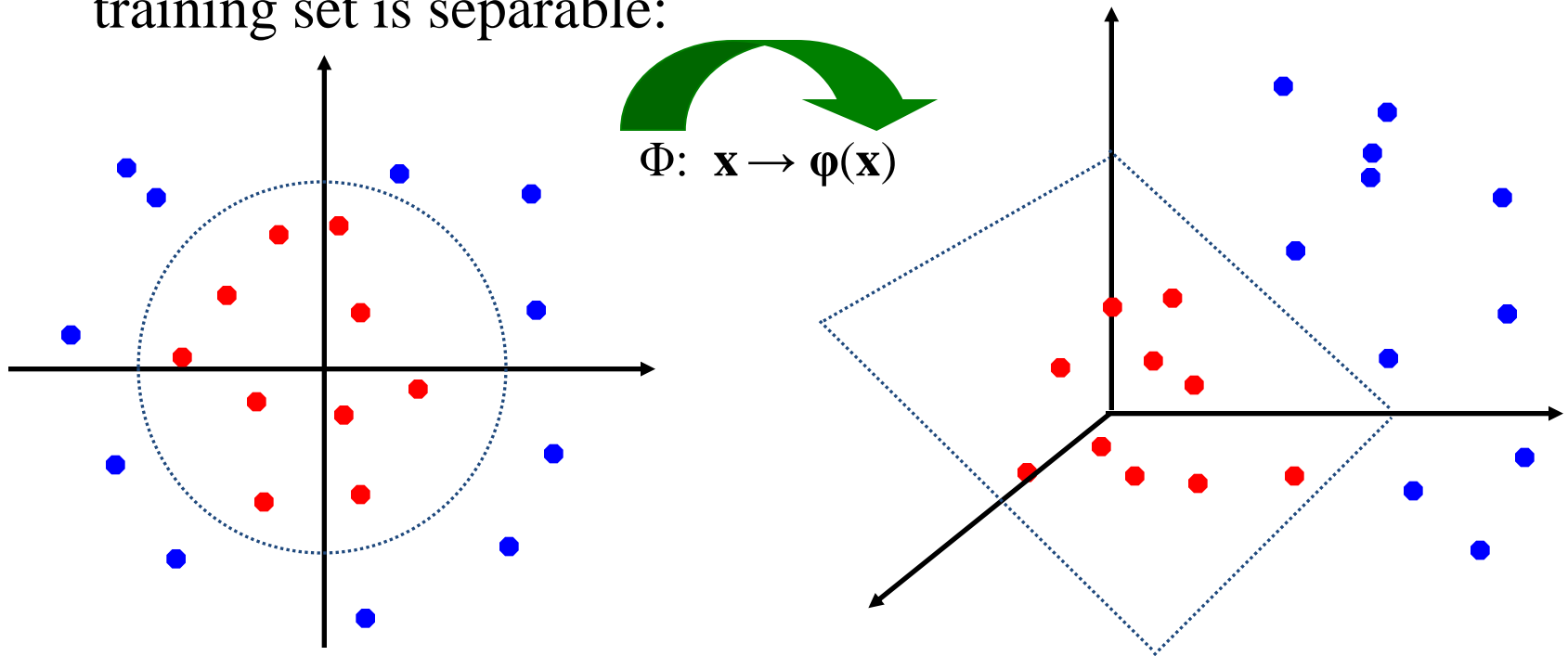
- Each non-zero α_i indicates that corresponding \mathbf{x}_i is a support vector.
- Then the classifying function is (note that we don't need \mathbf{w} explicitly):

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

- Notice that it relies on an *inner product* between the test point \mathbf{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 training points.

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:



Some Notes on the Parameter Setting

- Parameter Setting = Car Tuning
 - need much experience or many times of trial
 - you may get worse results if you are unlucky
- Multilayer Perceptron (MLP)
 - Main parameters for learning: hiddenLayers, learningRate, momentum, trainingTime (epoch), seed
- J48
 - Main parameters: unpruned, numFolds, minNumObj
 - Many parameters are for controlling the size of the result tree, i.e. confidenceFactor, pruning
- SMO (SVM)
 - Main parameters: c (complexity parameter), kernel, kernel parameters

Practice

- Basic
 - Comparing the **performances** of algorithms
 - MultilayerPerceptron vs. J48 vs. SVM
 - Checking the trained model (structure & parameter)
 - **Tuning** parameters to get **better models**
 - Understanding ‘**Test options**’ & ‘**Classifier output**’ in Weka
- Advanced
 - Building committee machines using ‘**meta**’ algorithms for classification
 - Preprocessing / data manipulation – applying ‘**Filter**’
 - Batch experiment with ‘**Experimenter**’
 - Design & run a batch process with ‘**KnowledgeFlow**’