Bayesian Classification: Why?

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem: Basics

- Let X be a data sample whose class label is unknown
- Let H be a hypothesis that X belongs to class C
- For classification problems, determine P(H/X): the probability that the hypothesis holds given the observed data sample X
- P(H): prior probability of hypothesis H (i.e. the initial probability before we observe any data, reflects the background knowledge)
- P(X): probability that sample data is observed
- P(X|H) : probability of observing the sample X, given that the hypothesis holds

Bayesian Theorem

- Given training data *X, posteriori probability of a hypothesis H, P(H|X)* follows the Bayes theorem

$$P(H|X)=\frac{P(X|H)P(H)}{P(X)}$$

- Informally, this can be written as

  posterior =likelihood x prior / evidence

- MAP (maximum posteriori) hypothesis

$$h_{MAP}\equiv\underset{h\in H}{\arg\max}\,P(h|D)=\underset{h\in H}{\arg\max}\,P(D|h)P(h).$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- The product of occurrence of say 2 elements x_1 and x_2, given the current class is C, is the product of the probabilities of each element taken separately, given the same class $P([y_1,y_2],C) = P(y_1,C) * P(y_2,C)$
- No dependence relation between attributes
- Greatly reduces the computation cost, only count the class distribution.
- Once the probability $P(X|C_i)$ is known, assign X to the class with maximum $P(X|C_i)*P(C_i)$

Training dataset

Class:
C1:buys_computer= 'yes'
C2:buys_computer= 'no'

Data sample
X =(age<=30,
Income=medium,
Student=yes
Credit_rating=
Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: Example

- Compute P(X/Ci) for each class

P(age="<30" | buys_computer="yes") = 2/9=0.222
P(age="<30" | buys_computer="no") = 3/5 =0.6
P(income="medium" | buys_computer="yes")= 4/9 =0.444
P(income="medium" | buys_computer="no") = 2/5 = 0.4
P(student="yes" | buys_computer="yes")= 6/9 =0.667
P(student="yes" | buys_computer="no")= 1/5=0.2
P(credit_rating="fair" | buys_computer="yes")=6/9=0.667
P(credit_rating="fair" | buys_computer="no")=2/5=0.4

X=(age<=30 ,income =medium, student=yes,credit_rating=fair)

P(X|Ci) : P(X|buys_computer="yes")= 0.222 x 0.444 x 0.667 x 0.667 =0.044
P(X|buys_computer="no")= 0.6 x 0.4 x 0.2 x 0.4 =0.019

P(X|Ci)*P(Ci) : P(X|buys_computer="yes") * P(buys_computer="yes")=0.028
P(X|buys_computer="no") * P(buys_computer="no")=0.007

X belongs to class "buys_computer=yes"

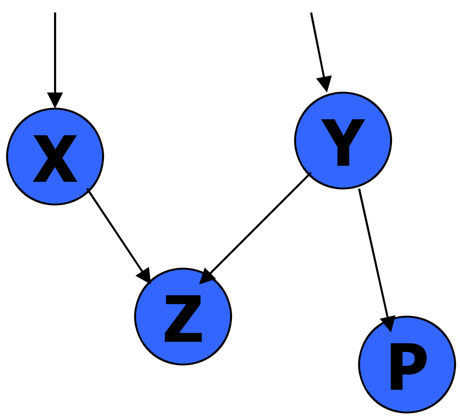
Naïve Bayesian Classifier: Comments

- Advantages :
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence , therefore loss of accuracy
  - Practically, dependencies exist among variables
  - E.g., hospitals: patients: Profile: age, family history etc

    Symptoms: fever, cough etc., Disease: lung cancer, diabetes etc
  - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

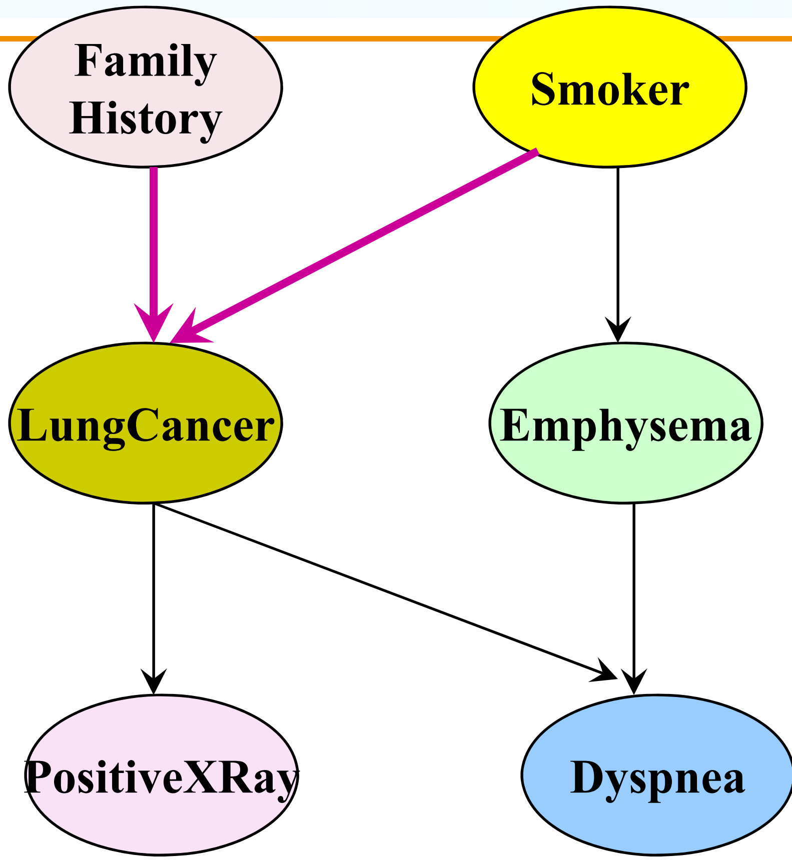
Bayesian Networks

- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



- ❑Nodes: random variables
- ❑Links: dependency
- ❑X,Y are the parents of Z, and Y is the parent of P
- ❑No dependency between Z and P
- ❑Has no loops or cycles

Bayesian Belief Network: An Example



Bayesian Belief Networks

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

The conditional probability table for the variable LungCancer: Shows the conditional probability for each possible combination of its parents

$$P(z1,\ldots,zn) = \prod_{i=1}^{n} P(z_i \mid Parents(Z_i))$$

Learning Bayesian Networks

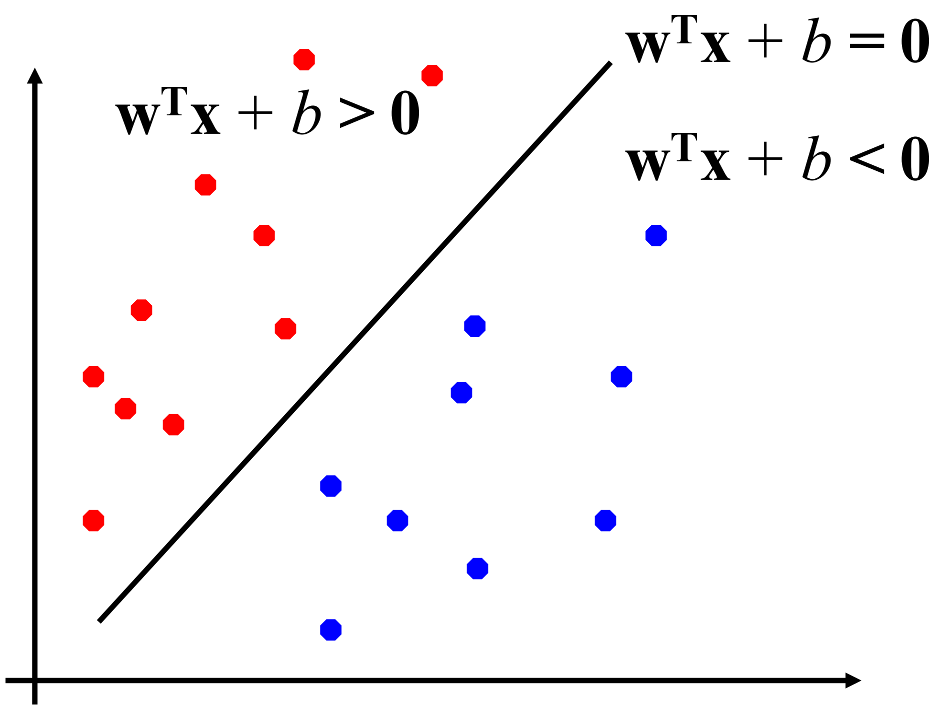
- Several cases
  - Given both the network structure and all variables observable: learn only the CPTs
  - Network structure known, some hidden variables: method of gradient descent, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to reconstruct graph topology
  - Unknown structure, all hidden variables: no good algorithms known for this purpose
- D. Heckerman, Bayesian networks for data mining

Support Vector Machines (SVM)

Oct 28, 2015

Linear Separators

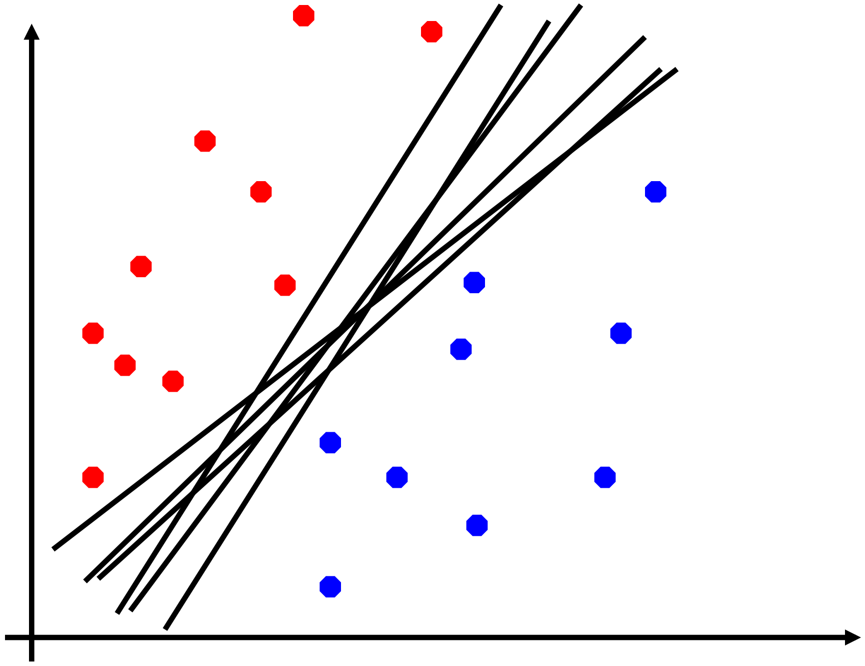
- Binary classification can be viewed as the task of separating classes in feature space:



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

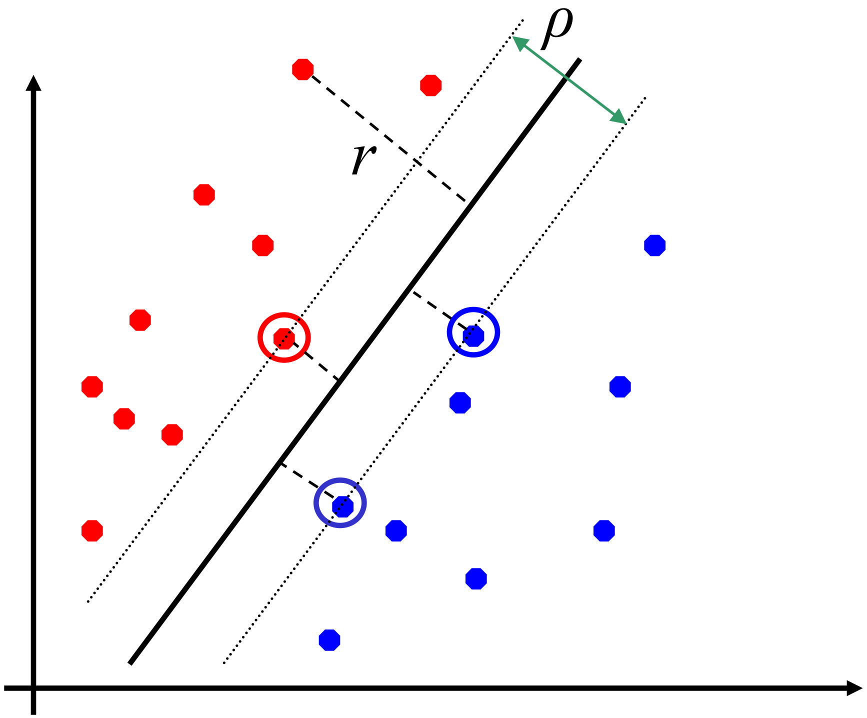
Linear Separators

- Which of the linear separators is optimal?



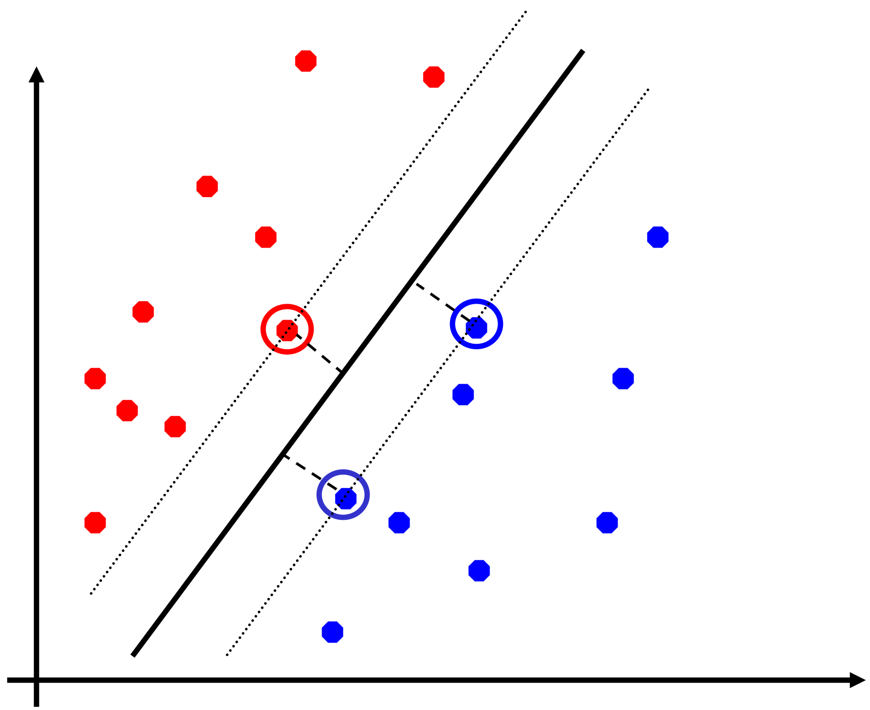
Classification Margin

- Distance from example $\mathbf{x}_i$ to the separator is $r = \dfrac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$
- 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
- 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 $(\mathbf{x}_i, y_i)$:

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

- 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 : $\quad y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1$

Which can be reformulated as:

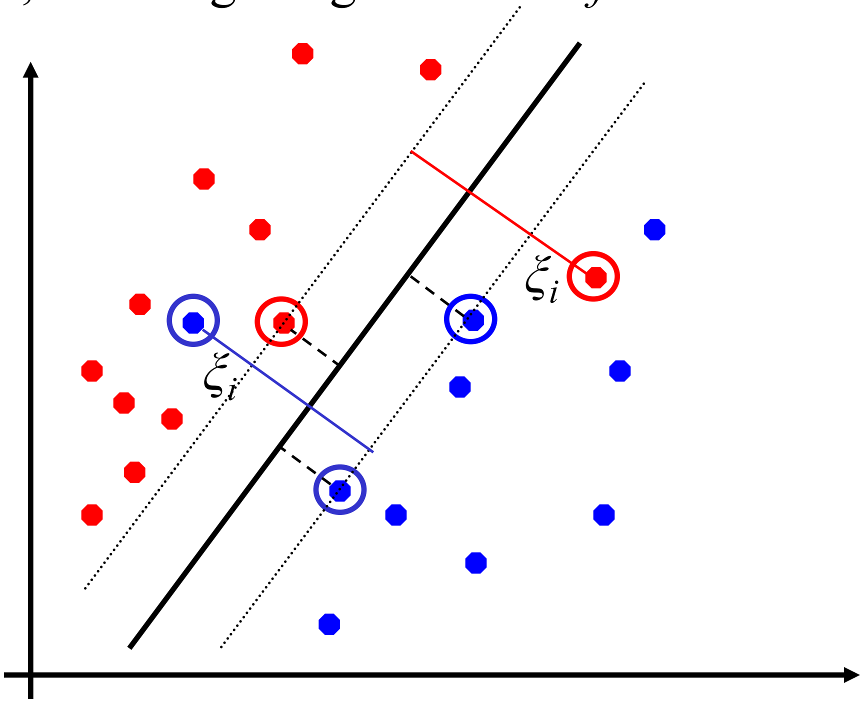
Find $\mathbf{w}$ and b such that

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

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

Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification Mathematically

- The old formulation:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, i=1..n : $y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1$

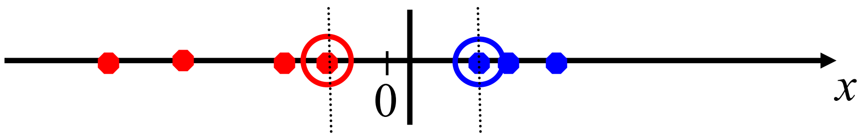
- Modified formulation incorporates slack variables:

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

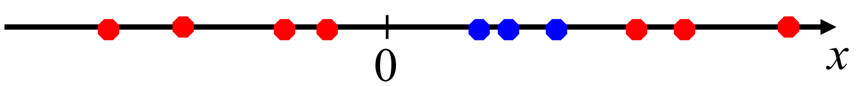
- Parameter C can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

Non-linear SVMs

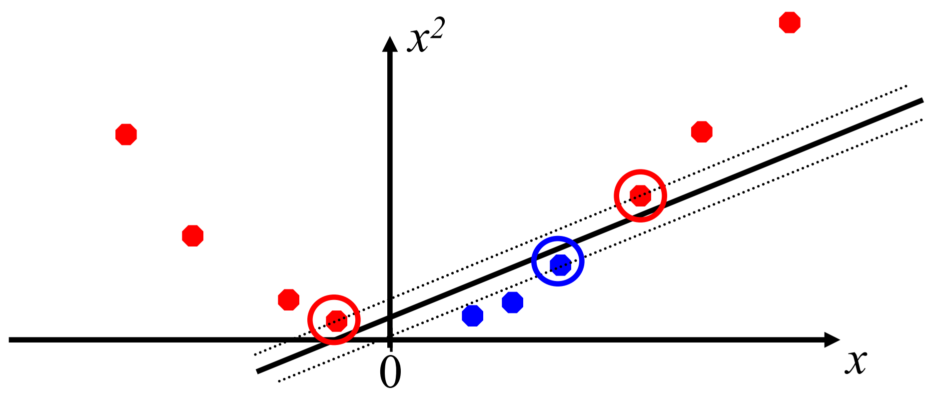
- Datasets that are linearly separable with some noise work out great:



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

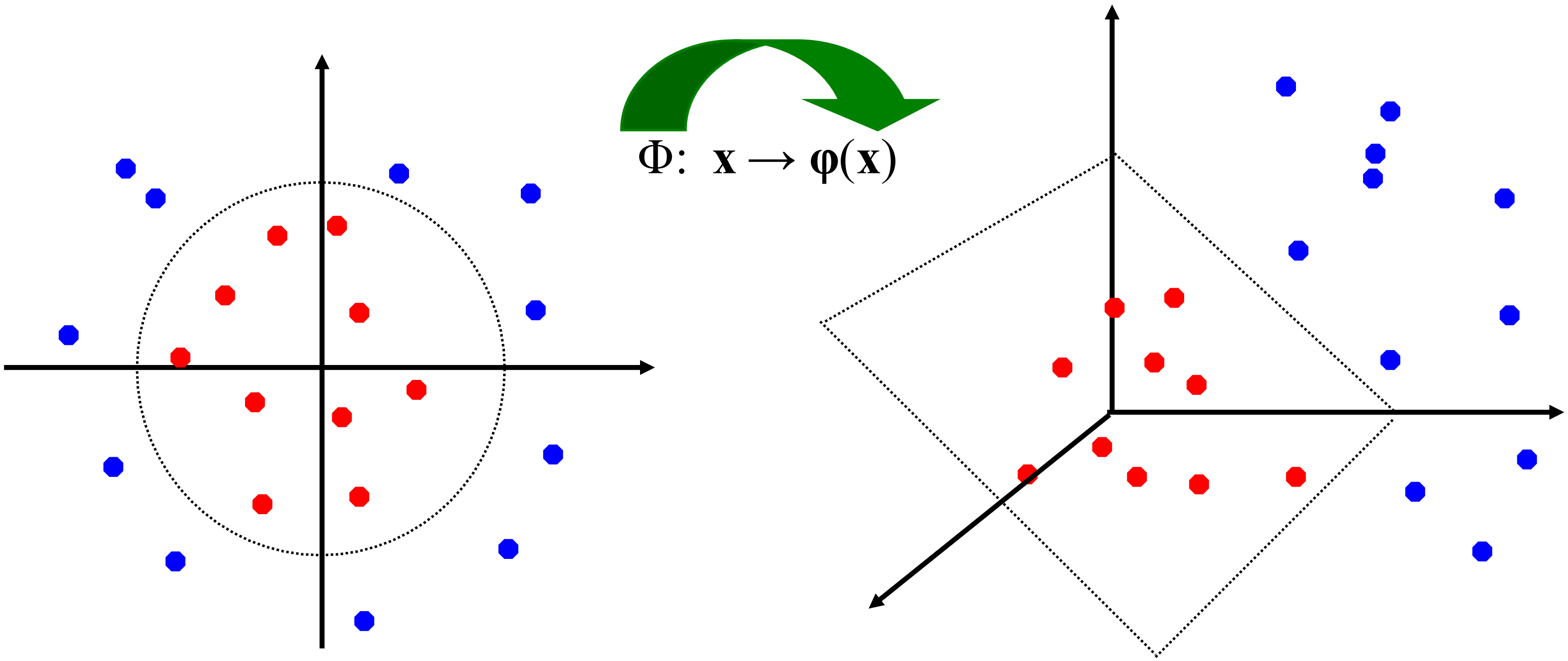


- How about… mapping data to a higher-dimensional space:



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:



SVM applications

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97], principal component analysis [Schölkopf *et al.* '99], etc.
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.

SVM slides reference: http://www.cs.utexas.edu/~mooney/cs391L/