Artificial Intelligence

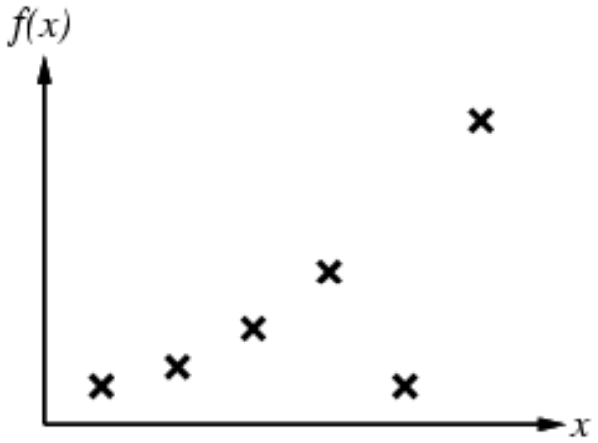
Fiona Yan Liu

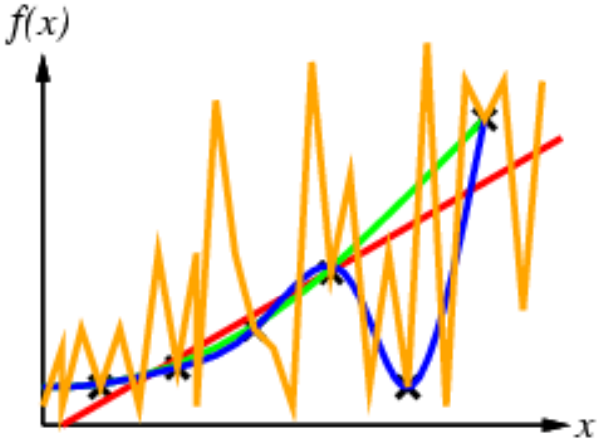
Department of Computing

The Hong Kong Polytechnic University

Inductive Learning

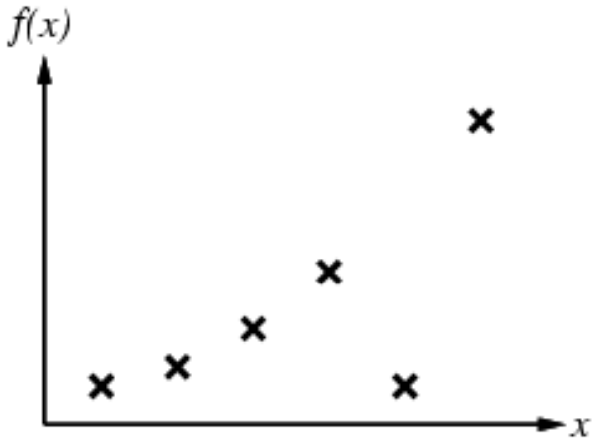
- Simplest form
  - learn a function *f* from examples pair (*x*, *f(x)*)
- Problem
  - Given a training set of examples
  - Find a hypothesis *h* such that $h \approx f$

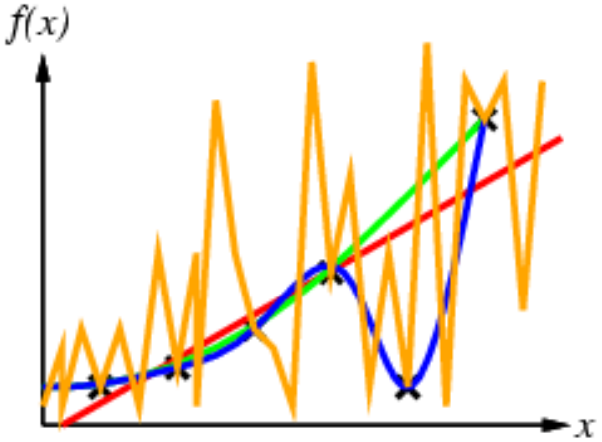




Inductive Learning

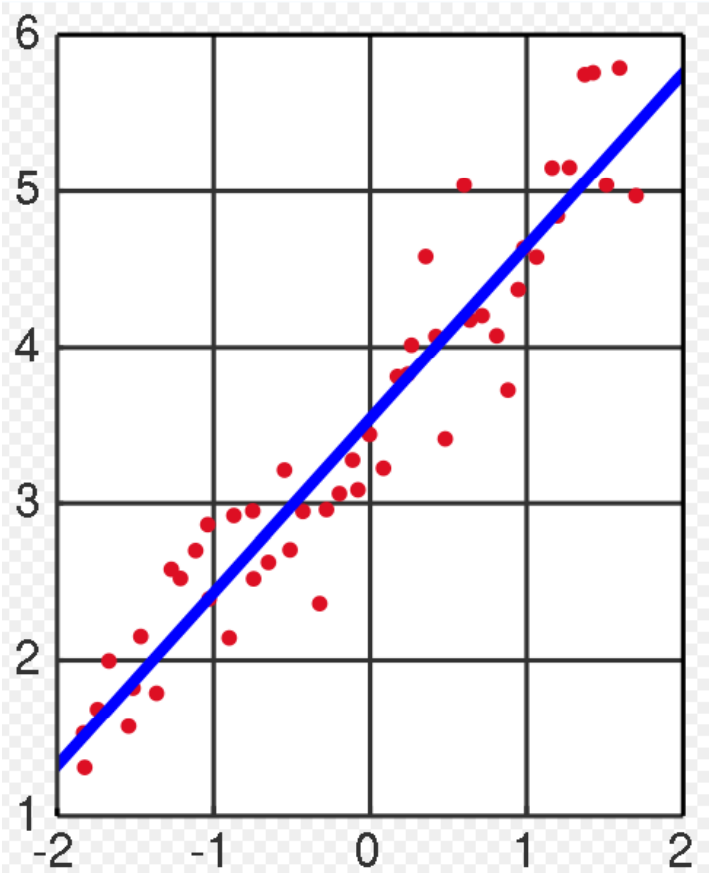
- How do we know that the hypothesis *h* is close to the target function *f* if we don't know what *f* is?
  - How many examples do we need to get a good h?
  - What hypothesis space should we use?
  - How complex should h be?
  - How do we avoid overfitting?





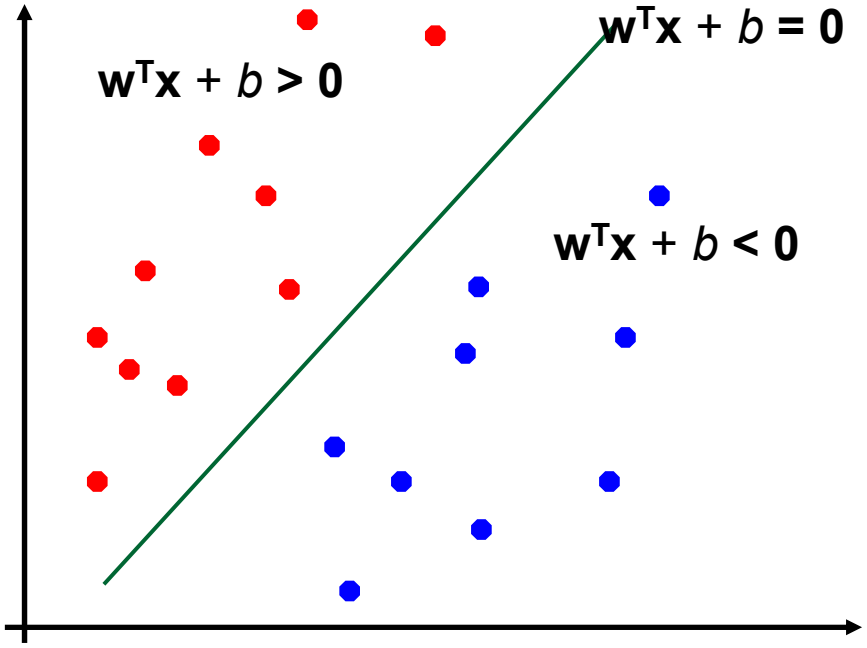
Univariate Linear Regression

- Regression with a univariate linear function is also known as "fitting a straight line".
- A univariate linear function (a straight line) with input x and output y has the form $y = w_1 x + w_0$
- Loss function: $Loss(h_w) = \sum_j (y_j - (w_1 x_j + w_0))$



Linear Classification

- Linear classification can be viewed as the task of finding the linear separator that can separate different classes in the feature space:



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

Probably Approximately Correct Learning

- Any hypothesis that is seriously wrong will almost certainly be found out with high probability after a small number of examples
  - It will make an incorrect prediction.
- Any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong
  - It must be probably approximately correct
- Probably approximately correct learning algorithm
  - Any learning algorithm that returns hypotheses that are probably approximately correct

Correctness of Hypothesis

- A hypothesis is called approximately correct if $error(h) \leq \varepsilon$, where ε is a small constant
  - $error(h)$: the error rate of a hypothesis
- For a "seriously wrong" hypothesis h_b belonging to H_{bad}, we have $error(h_b) > \varepsilon$, so the probability that it agrees with the first N examples is

$$P(h_b \text{ agrees with } N \text{ examples}) \leq (1-\varepsilon)^N$$

- The probability that H_{bad}, contains at least one consistent hypothesis is bounded by

$$P(H_{bad} \text{ contains a consistent hypothesis}) \leq |H_{bad}|(1-\varepsilon)^N \leq |H|(1-\varepsilon)^N$$

- We would like the probability of this event below some small number δ:

$$|H|(1-\varepsilon)^N \leq \delta$$

- Given that $1-\varepsilon \leq e^{-\varepsilon}$, we can achieve this if we allow the algorithm to see

$$N \geq \ln(|H|/\delta)/\varepsilon$$

Examples

- This number N, as a function of ε and δ, is called the sample complexity of the hypothesis space

How to Choose the Hypothesis space

- To obtain real generalization to unseen examples, then, it seems we need to restrict the hypothesis space.
- Several ways to restrict the hypothesis space
  - to bring prior knowledge
  - to insist that the algorithm return not just any consistent hypothesis, but preferably a simple one
  - to focus on learnable subsets of the entire hypothesis space

Linear Classification with Hard Threshold

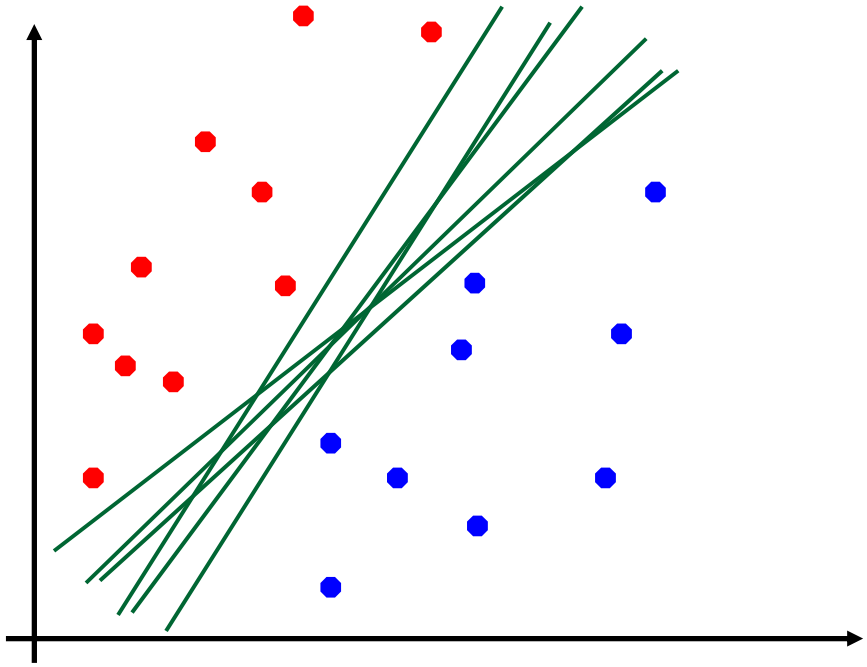
- Linear functions can be used to do classification as well as regression:
- $h_{\mathbf{w}}(\mathbf{x}) = \text{Threshold}(\mathbf{w}^T\mathbf{x})$
- where Threshold(z)= 1 if $z > 0$ and 0 otherwise.
- Since the loss function is undifferentiable, we cannot obtain the solution as in the regression problem.
- $w_i \leftarrow w_i + \alpha(y\text{-}h_w(\mathbf{x}))x_i$

Linear Classification with Logistic Regression

- The problems of linear classification with hard threshold:
  - $h_w(\mathbf{x})$ is not differentiable
  - Linear classifier always announces a completely confident prediction of 1 or 0 even for examples close to the boundary
- $h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w}^T\mathbf{x}) = 1/(1+e^{-\mathbf{w}^T\mathbf{x}})$
- $w_i \leftarrow w_i + \alpha(y-h_w(\mathbf{x}))(1-h_w(\mathbf{x}))h_w(\mathbf{x})x_i$

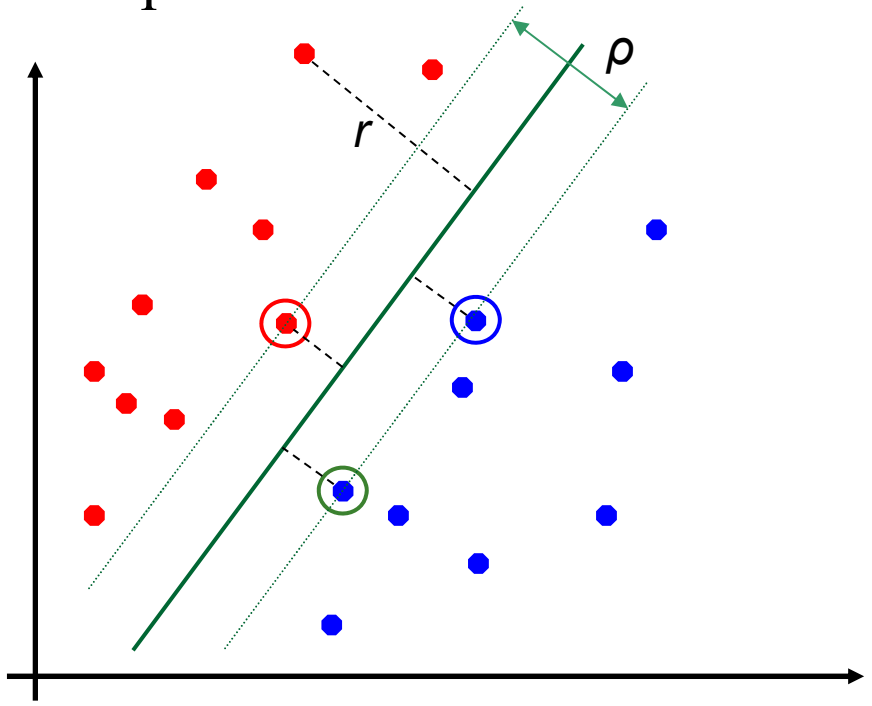
Multiple Solutions

- Which of the linear separators is the best?



Classification Margin

- Distance from example $\mathbf{x}_i$ to the separator is $r = |(\mathbf{w}^T\mathbf{x}_i + b)| / \|\mathbf{w}\|$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.



Example of SVM

- Given two support vectors
  - (5, 1) belonging to class 1
  - (-1, -1) belonging to class 2
- Calculate w_1 and w_0
  - Make the linear hyperplane $y = w_1x + w_0$ give the maximum margin
- **$y = -3x+6$**

The kernel function

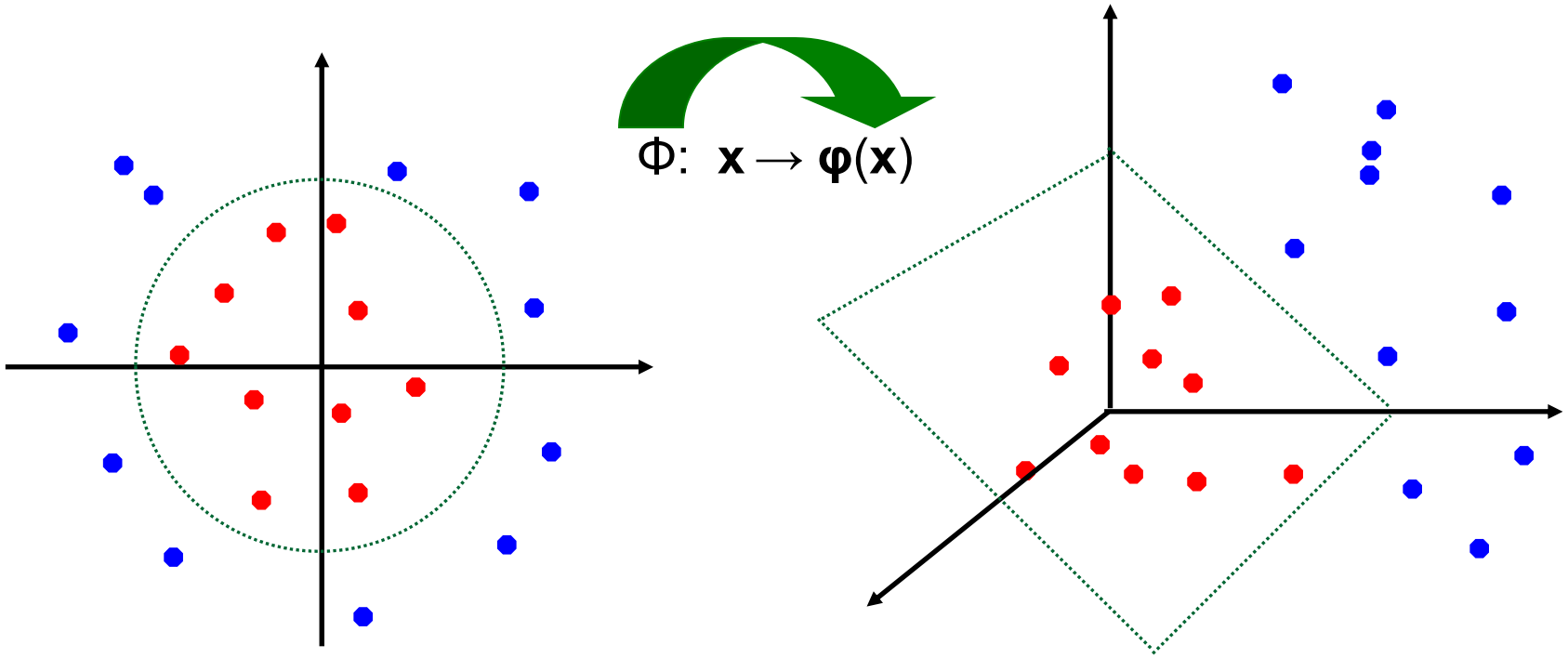
- The linear classifier relies on inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

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

- A *kernel function* is a function that is eqiuvalent to an inner product in some feature space. Every semi-positive definite symmetric function is a kernel.
- A kernel function implicitly maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).

Nonlinear SVM

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



The Kernel Function

- The linear classifier relies on inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:
  - $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$
- A *kernel function* is a function that is eqiuvalent to an inner product in some feature space. Every semi-positive definite symmetric function is a kernel.
- A kernel function implicitly maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).

Examples of Kernel Functions

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

Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself

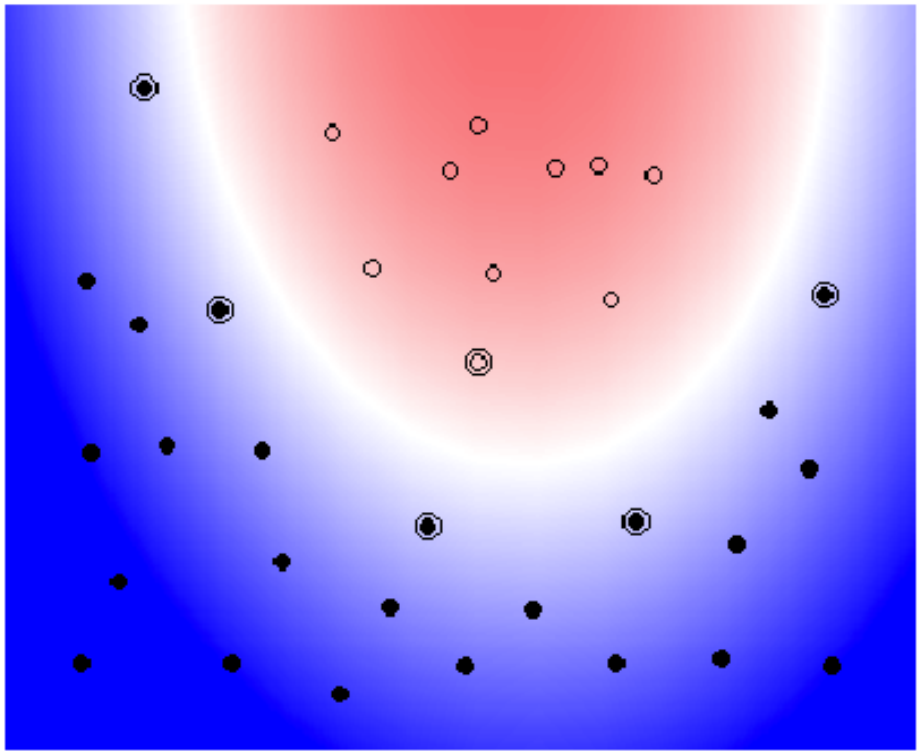
Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^p$

Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

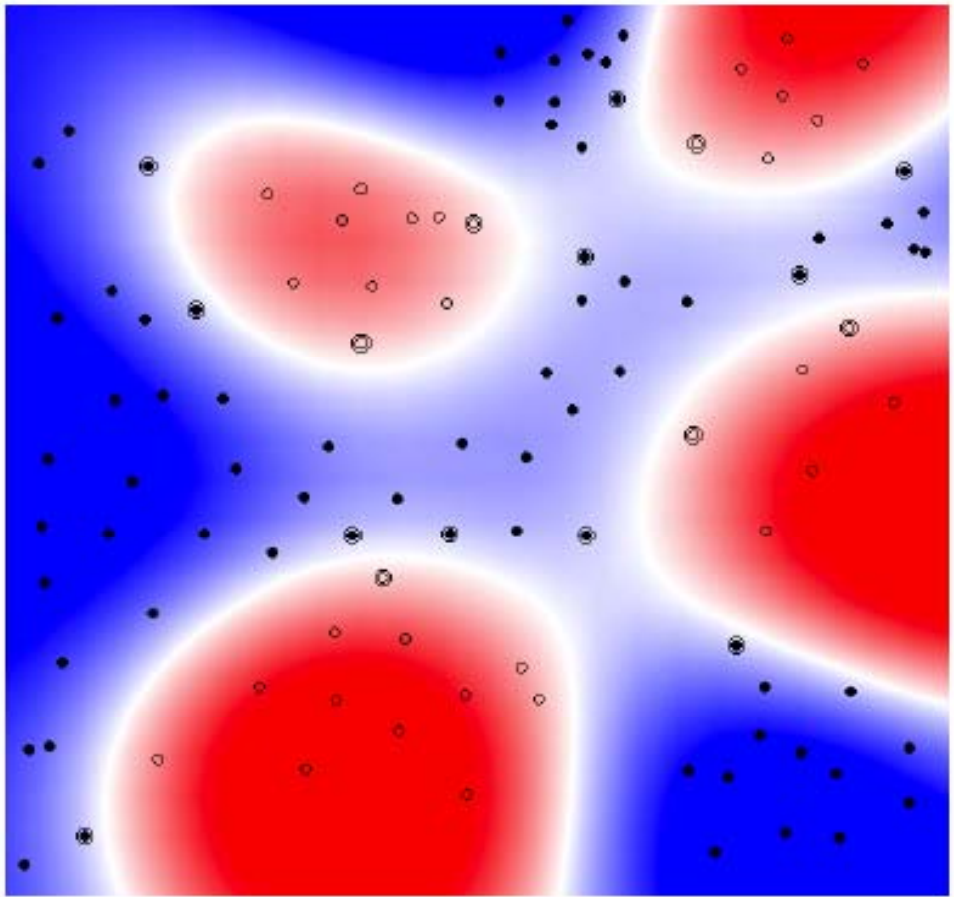
Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$

Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

Examples of SVM with Polynomial Kernel



Examples of SVM with Radial-basis Function Kernel



Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
  - Only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces
  - Complexity does not depend on the dimensionality of the feature space
- Nice math property
  - A simple convex optimization problem which is guaranteed to converge to a single global solution