Classification / Regression

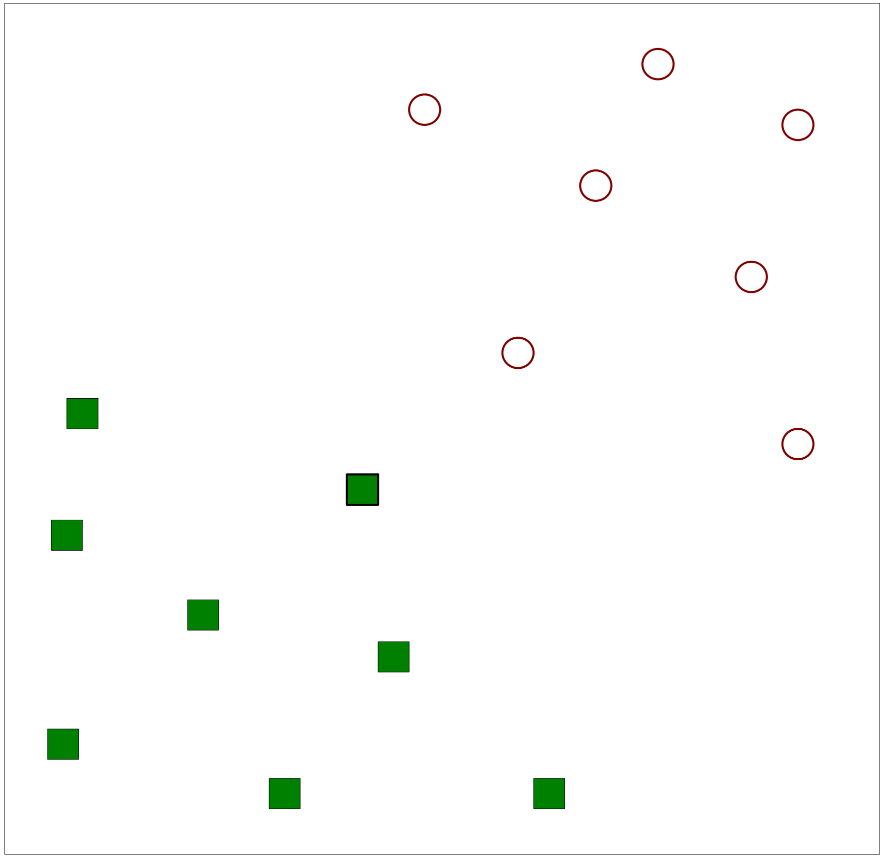
Support Vector Machines

Support vector machines

- Topics
  - SVM classifiers for linearly separable classes
  - SVM classifiers for non-linearly separable classes
  - SVM classifiers for nonlinear decision boundaries
    - kernel functions
  - Other applications of SVMs
  - Software

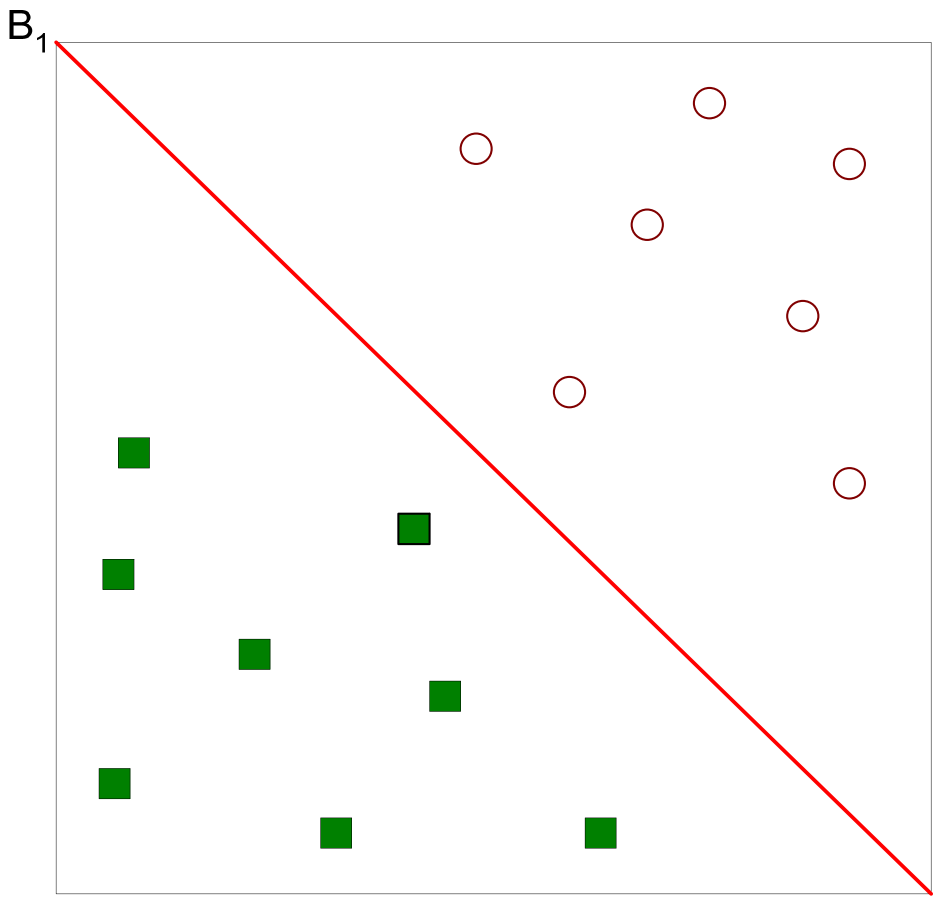
Support vector machines

Linearly separable classes



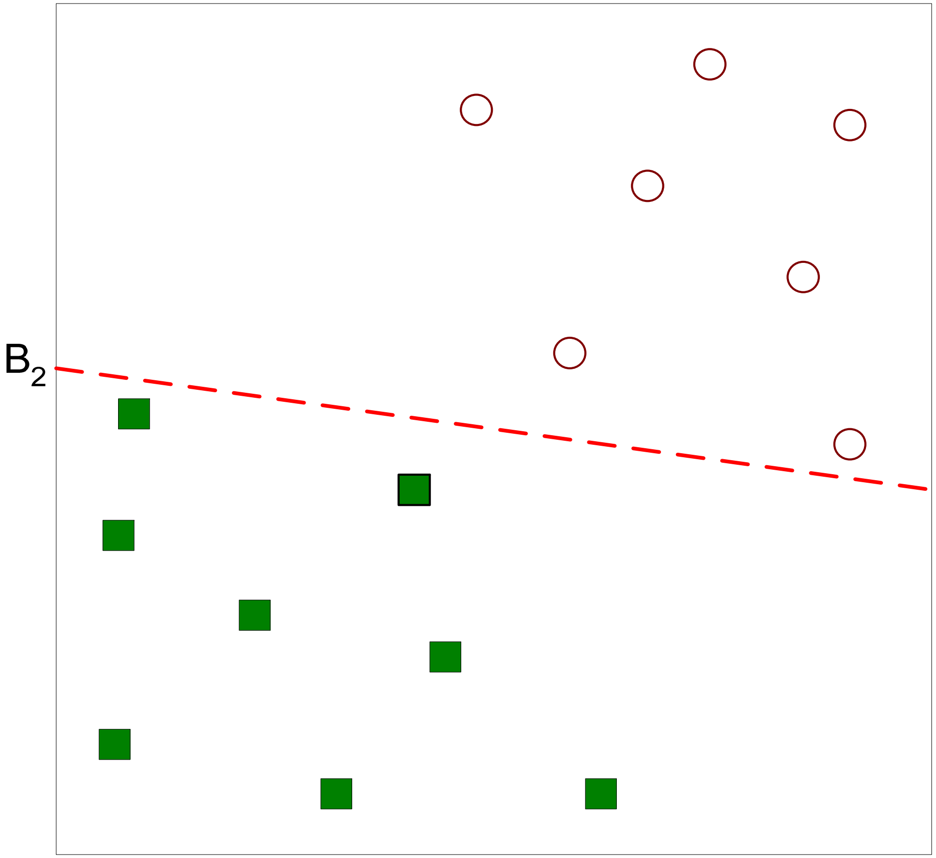
Goal: find a linear decision boundary (hyperplane) that separates the classes

Support vector machines



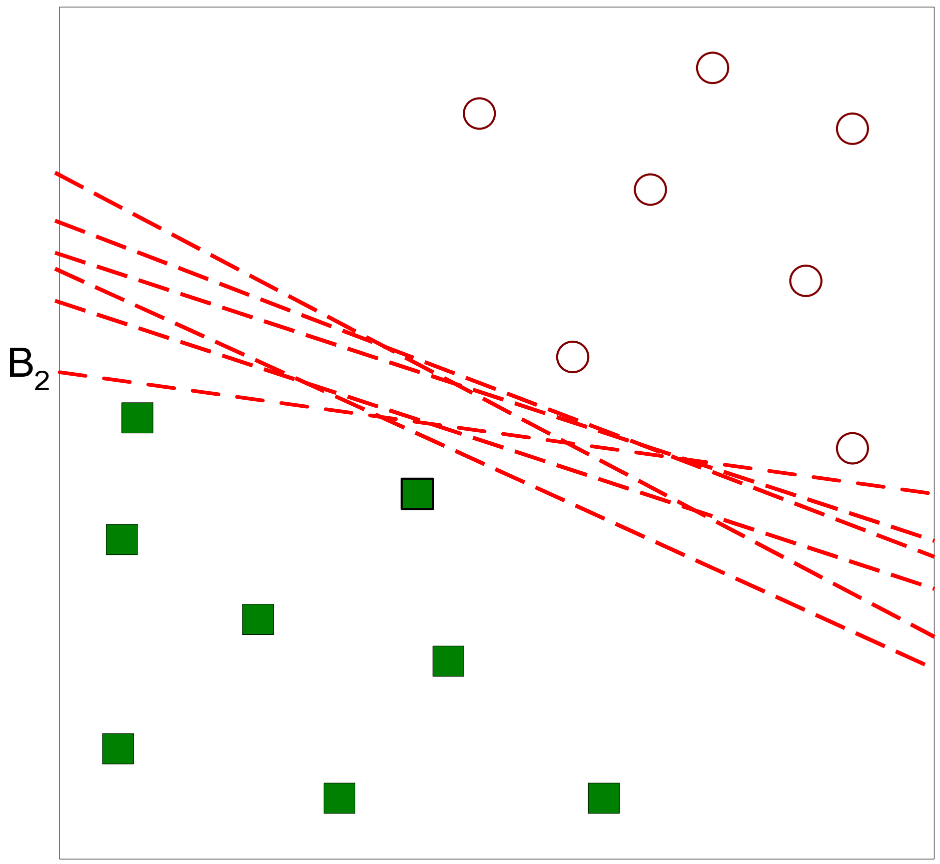
One possible solution

Support vector machines



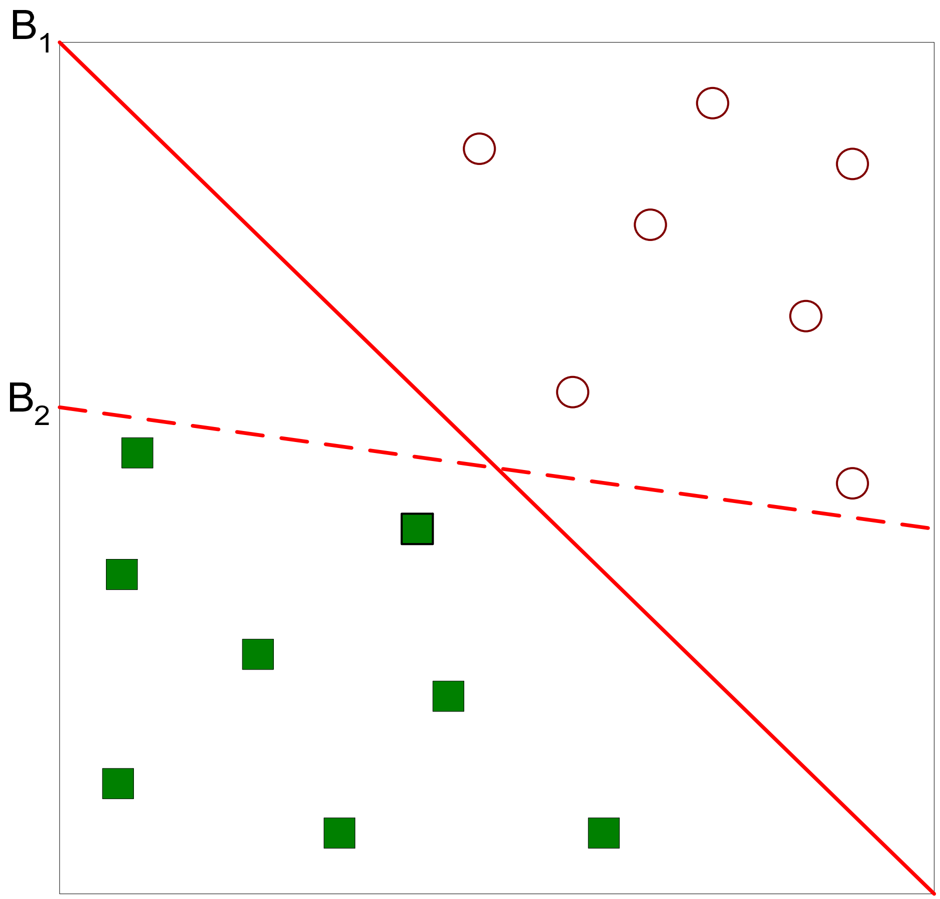
Another possible solution

Support vector machines



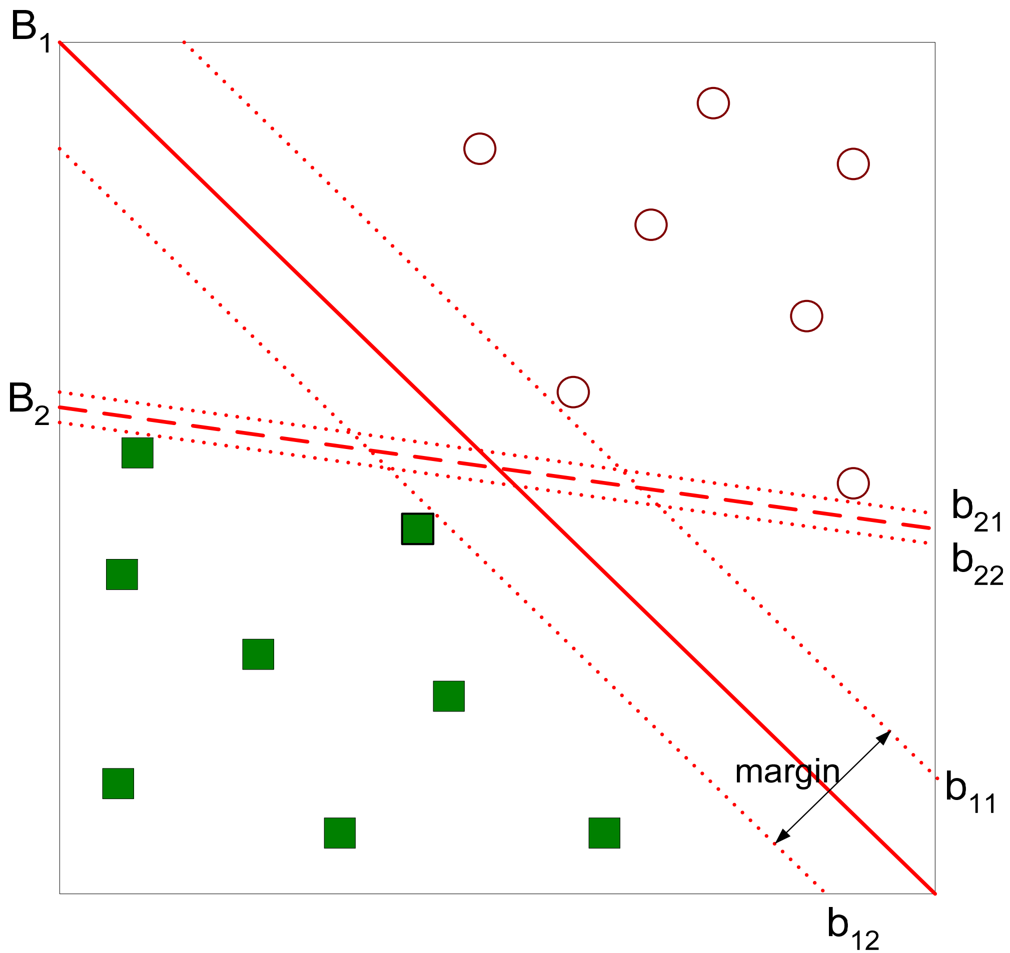
Other possible solutions

Support vector machines



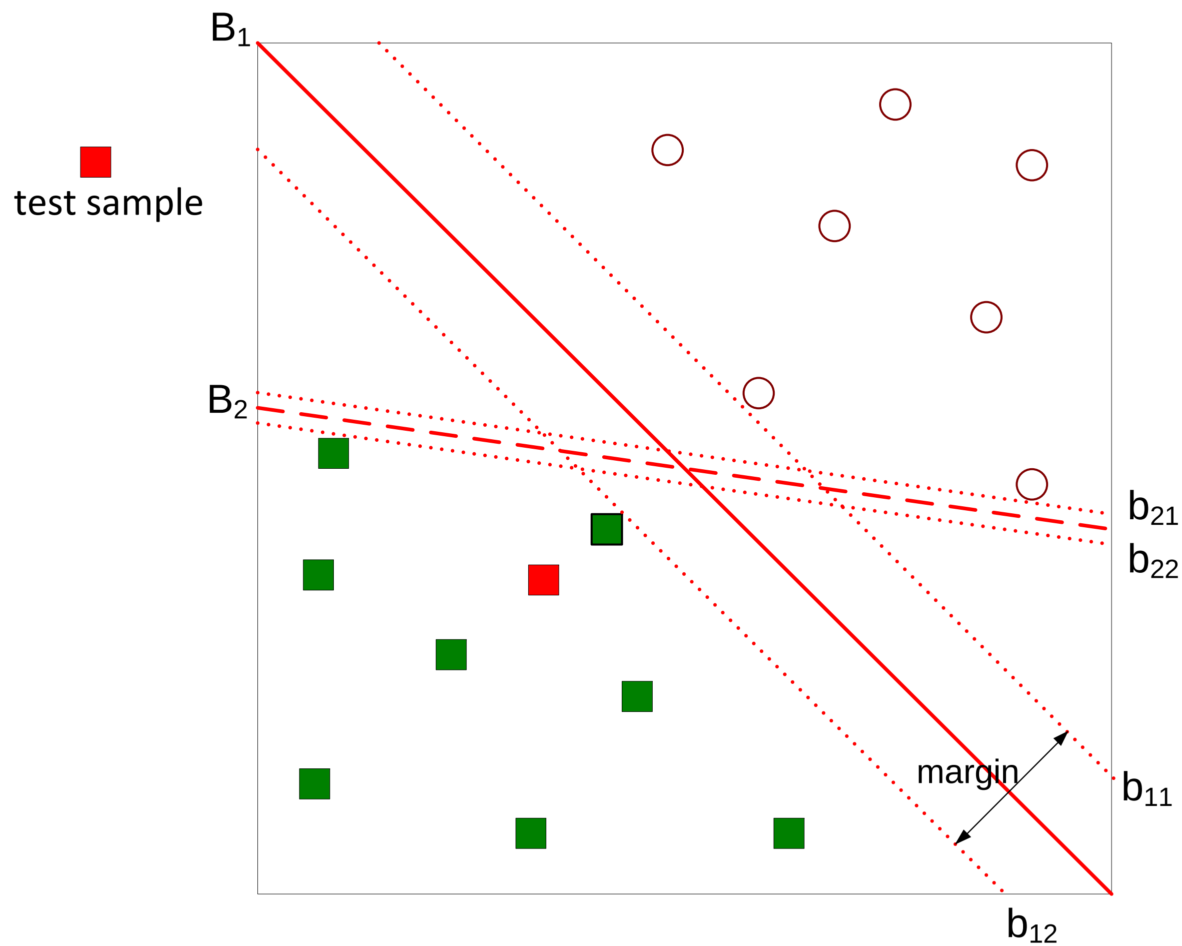
Which one is better? B1 or B2? How do you define better?

Support vector machines



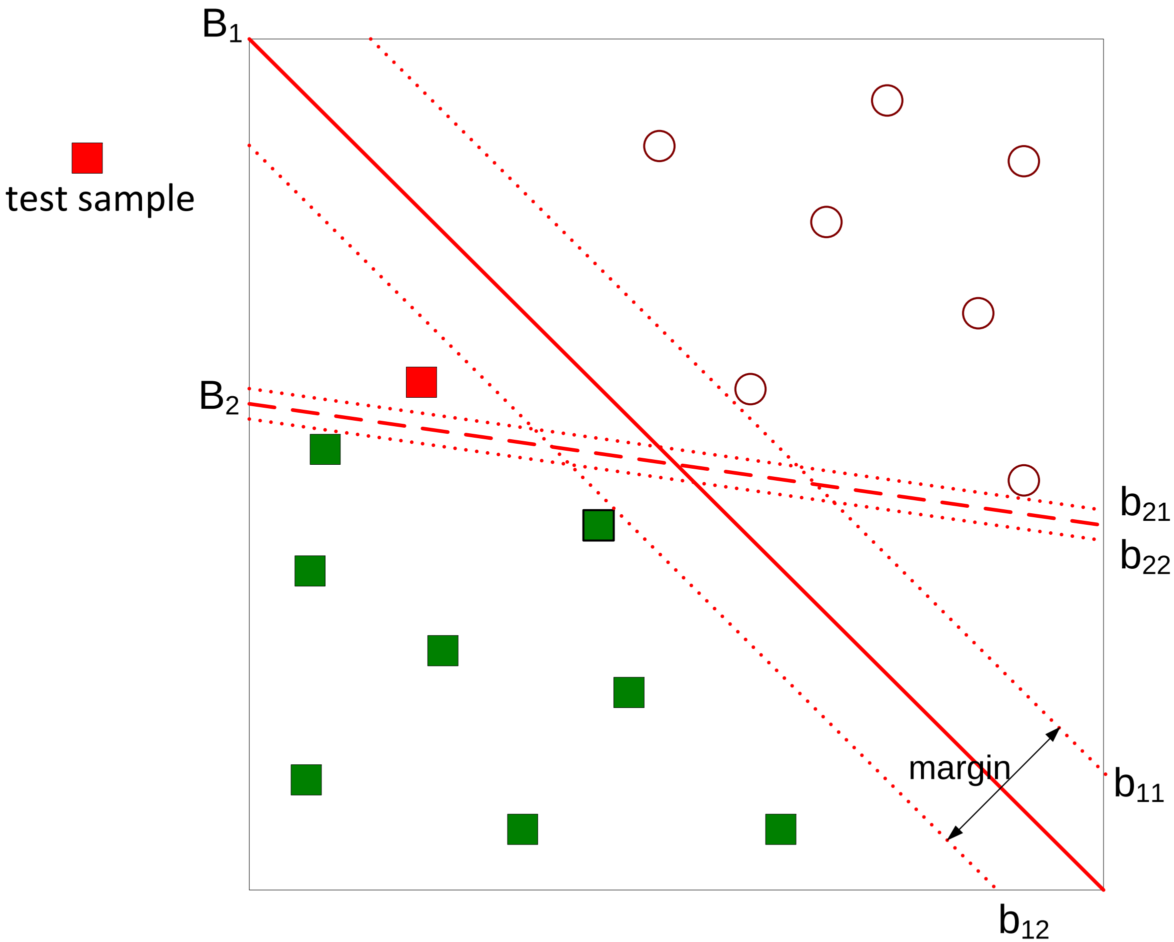
Hyperplane that maximizes the margin will have better generalization
=> B1 is better than B2

Support vector machines



Hyperplane that maximizes the margin will have better generalization
=> B1 is better than B2

Support vector machines

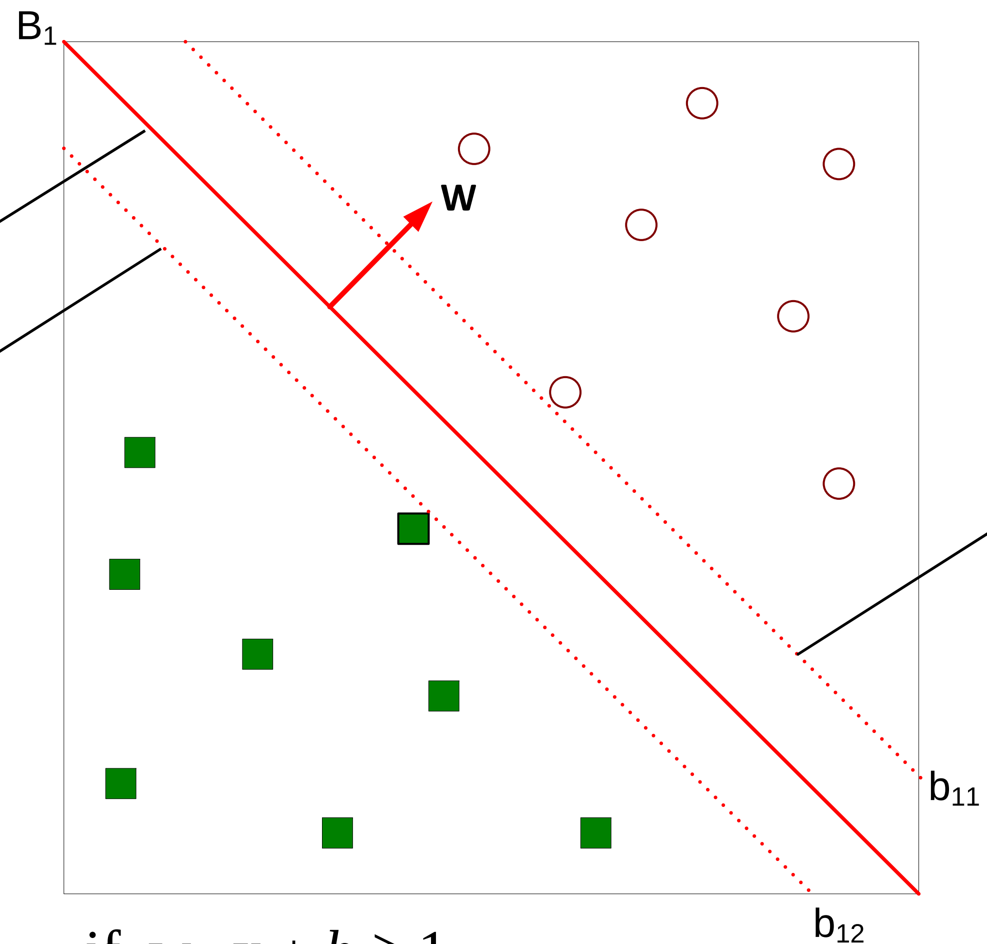


Hyperplane that maximizes the margin will have better generalization
=> B1 is better than B2

Support vector machines

$$\mathbf{w}\cdot\mathbf{x}+b=0$$

$$\mathbf{w}\cdot\mathbf{x}+b=-1$$



$$\mathbf{w}\cdot\mathbf{x}+b=+1$$

$$y_i = f(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}\cdot\mathbf{x}+b \geq 1 \\ -1 & \text{if } \mathbf{w}\cdot\mathbf{x}+b \leq -1 \end{cases}$$

$$\text{margin} = \frac{2}{\|\mathbf{w}\|}$$

Support vector machines

- We want to maximize: $\text{margin} = \dfrac{2}{\|\mathbf{w}\|}$
- Which is equivalent to minimizing: $L(\mathbf{w}) = \dfrac{\|\mathbf{w}\|^2}{2}$
- But subject to the following constraints:

$$y_i = f(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 1 \\ -1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq -1 \end{cases}$$

  - This is a constrained convex optimization problem
  - Solve with numerical approaches, e.g. quadratic programming

Support vector machines

Solving for **w** that gives maximum margin:

1. Combine objective function and constraints into new objective function, using Lagrange multipliers λ_i

$$L_{primal} = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left(y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right)$$

2. To minimize this Lagrangian, we take derivatives of **w** and *b* and set them to 0:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

Support vector machines

Solving for **w** that gives maximum margin:

3. Substituting and rearranging gives the dual of the Lagrangian:

$$L_{dual} = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

   which we try to maximize (not minimize).

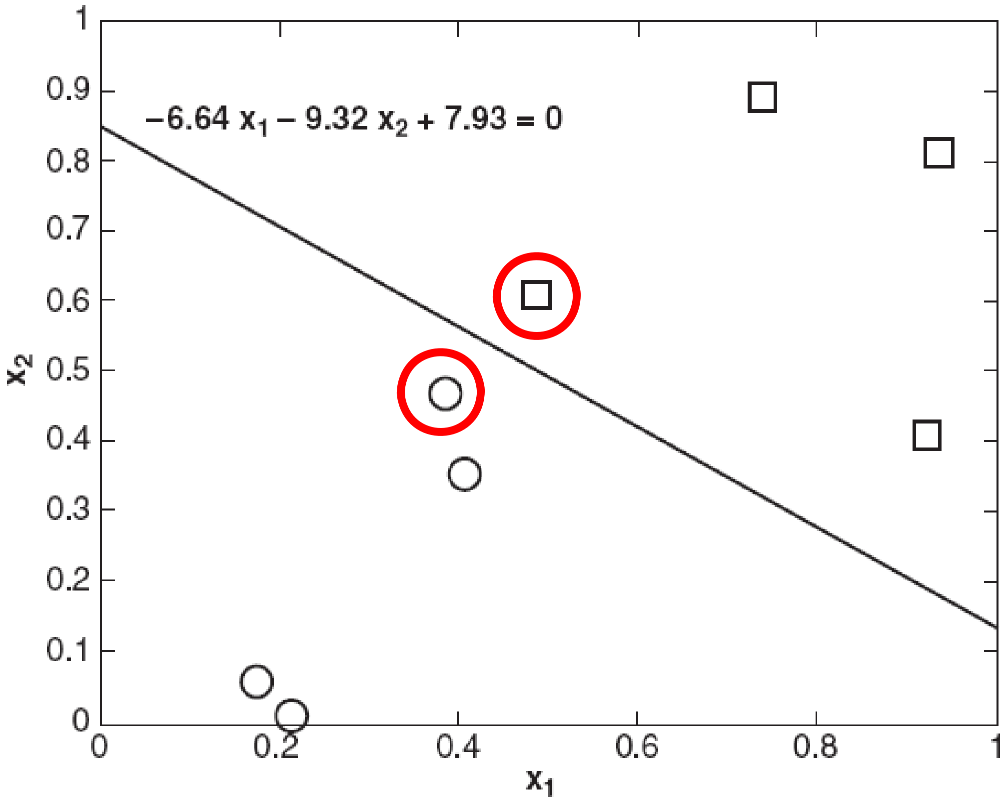
4. Once we have the λ_i, we can substitute into previous equations to get **w** and *b*.
5. This defines **w** and *b* as linear combinations of the training data.

Support vector machines

- Optimizing the dual is easier.
  - Function of λ_i only, not λ_i and **w**.
- Convex optimization $\Rightarrow$ guaranteed to find global optimum.
- Most of the λ_i go to zero.
  - The $\mathbf{x}_i$ for which $\lambda_i \neq 0$ are called the support vectors. These "support" (lie on) the margin boundaries.
  - The $\mathbf{x}_i$ for which $\lambda_i = 0$ lie away from the margin boundaries are not required for defining the maximum margin hyperplane.

Support vector machines

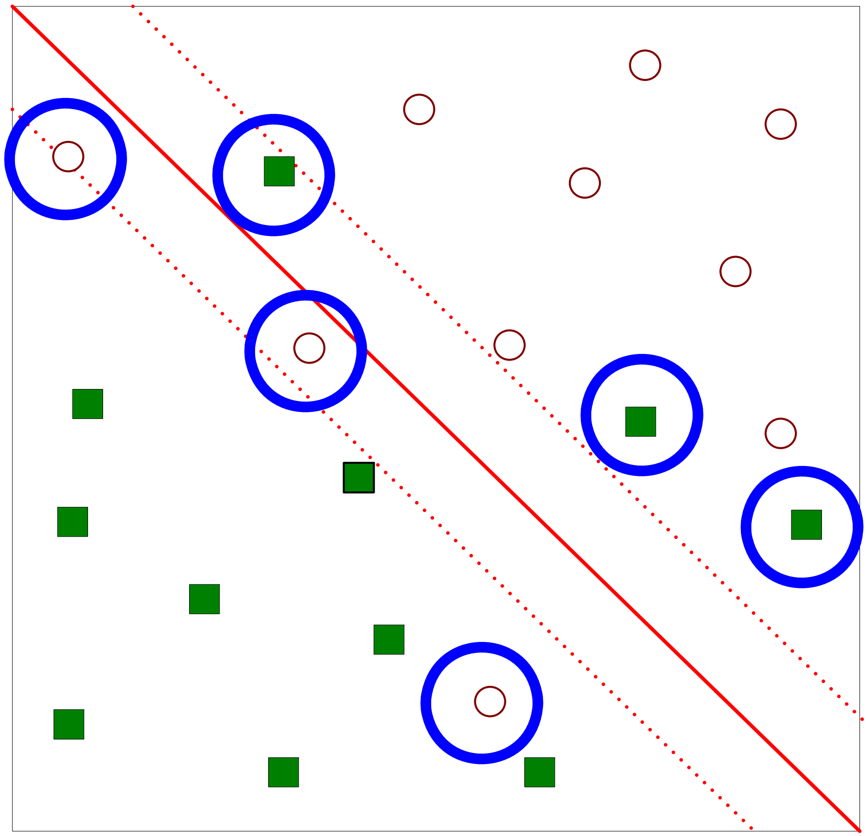
Example of solving for maximum margin hyperplane



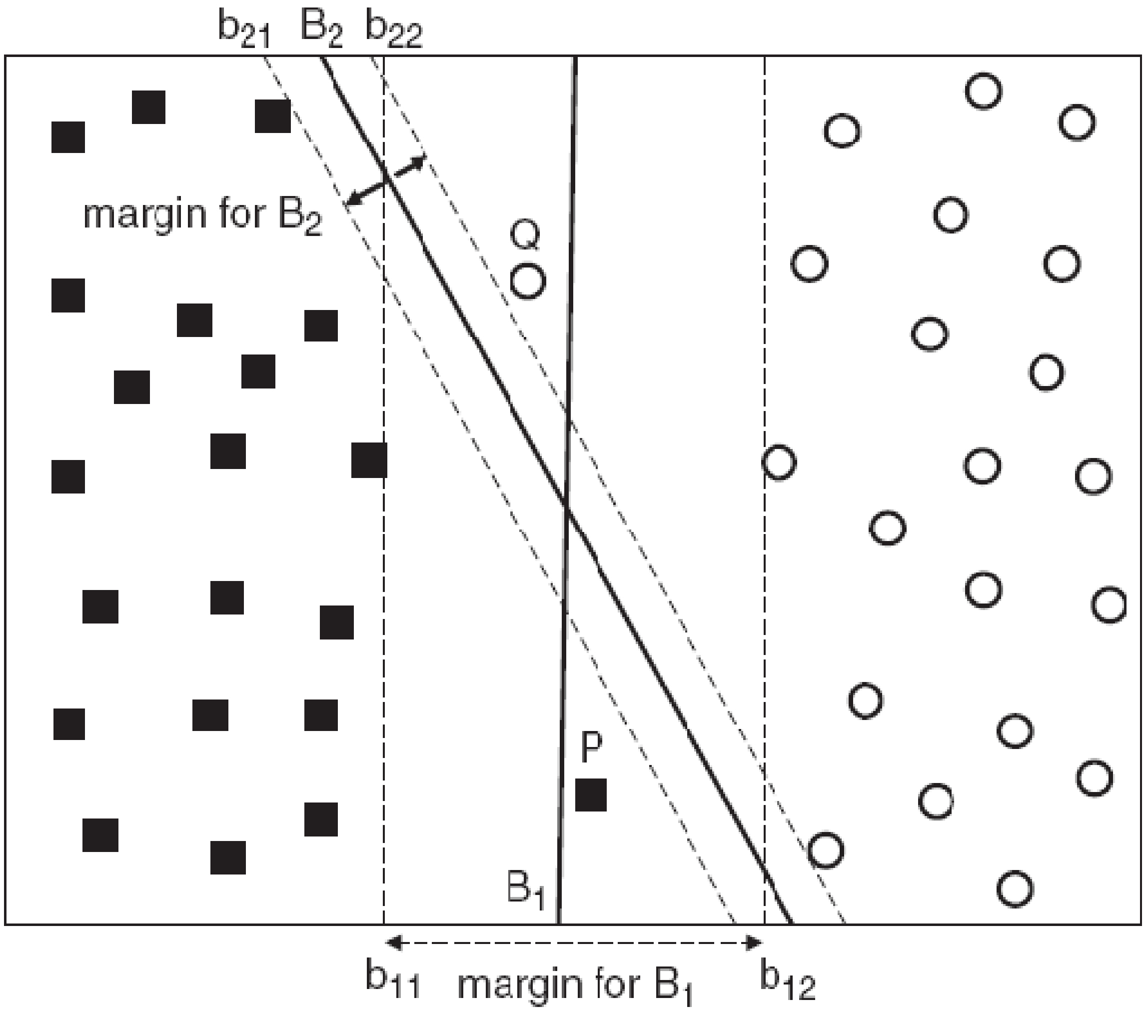
x_1	x_2	y	Lagrange Multiplier
0.3858	0.4687	1	65.5261
0.4871	0.611	−1	65.5261
0.9218	0.4103	−1	0
0.7382	0.8936	−1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	−1	0
0.2146	0.0099	1	0

Support vector machines

What if the classes are not linearly separable?



Support vector machines



Now which one is better? B1 or B2? How do you define better?

Support vector machines

- What if the problem is not linearly separable?
- Solution: introduce slack variables
  - Need to minimize:

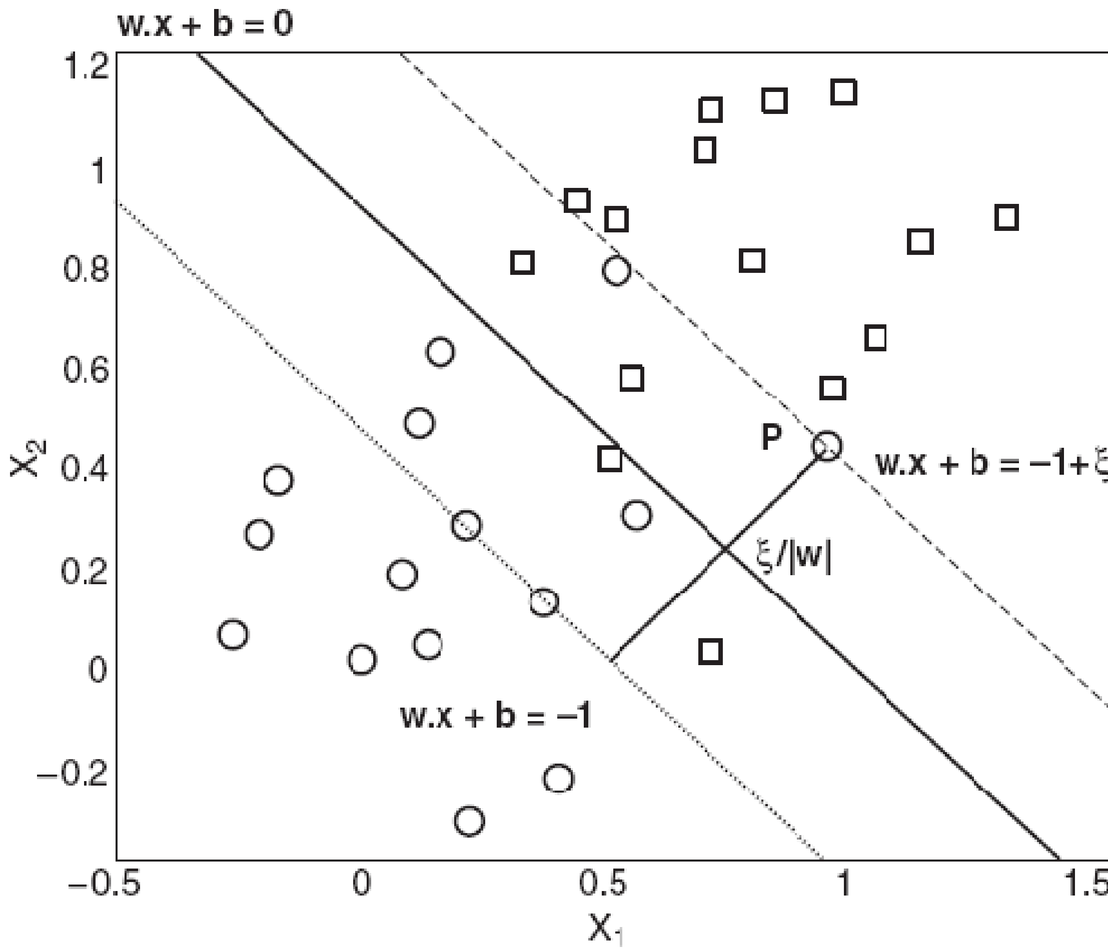
$$L(\mathbf{w}) = \frac{\| \mathbf{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k \right)$$

  - Subject to:

$$y_i = f(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 1 + \xi_i \\ -1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq -1 + \xi_i \end{cases}$$

  - C is an important hyperparameter, whose value is usually optimized by cross-validation.

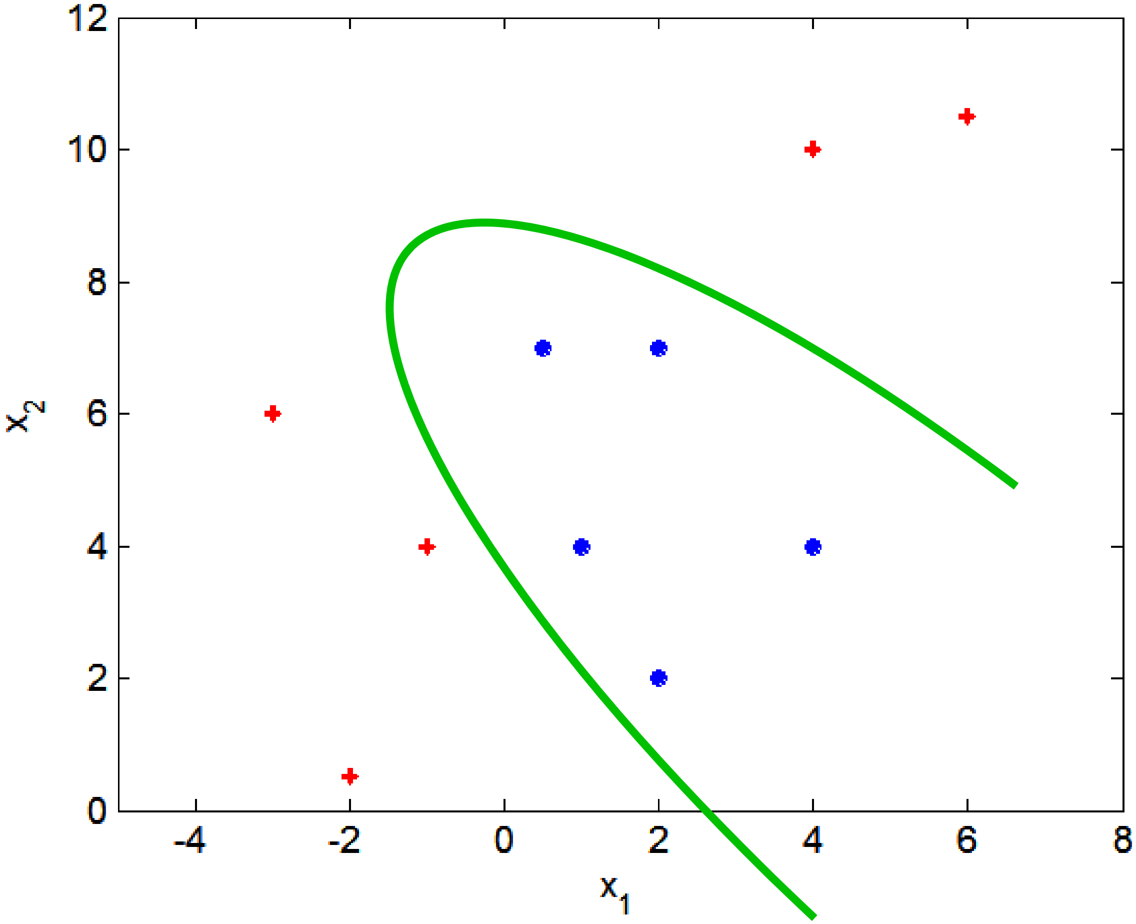
Support vector machines



Slack variables for nonseparable data

Support vector machines

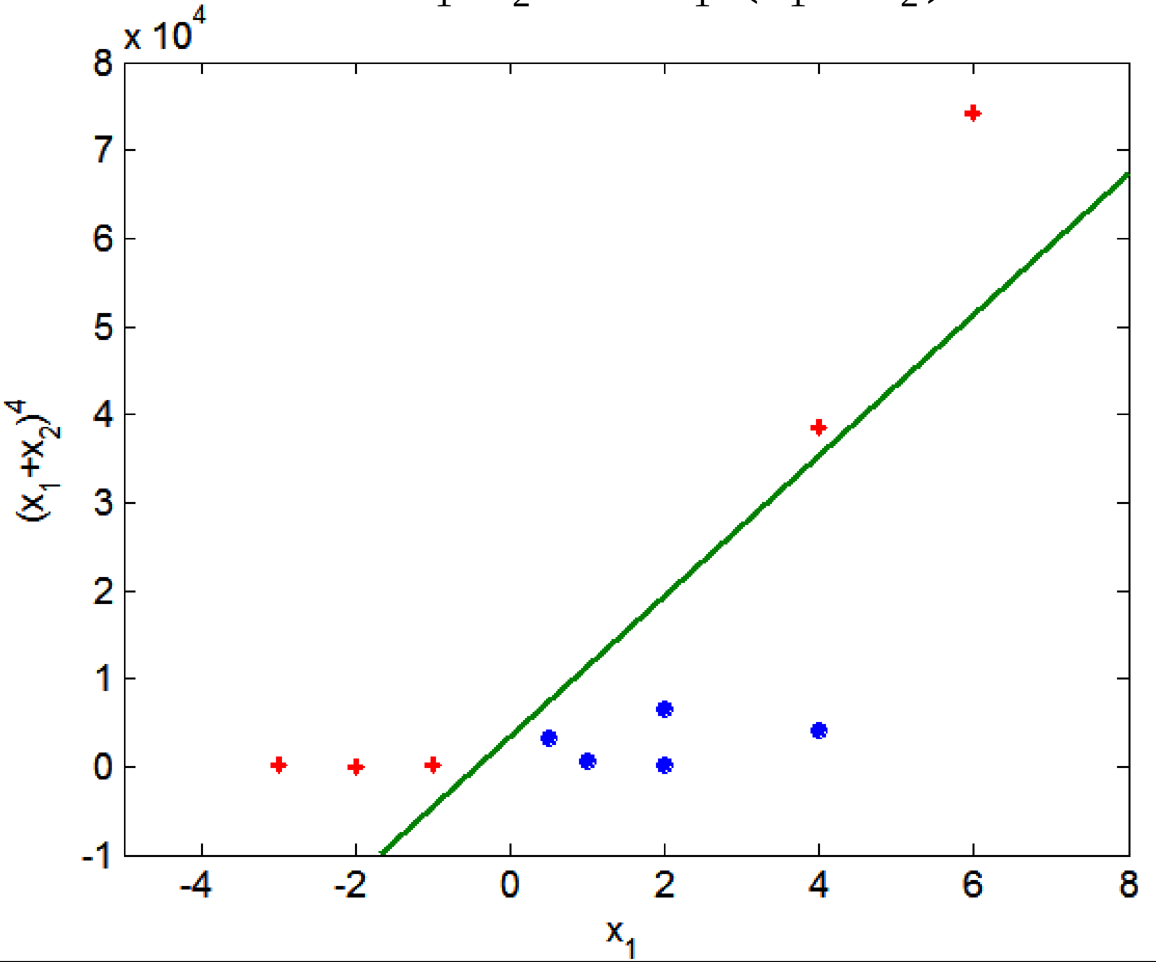
What if decision boundary is not linear?



Support vector machines

Solution: nonlinear transform of attributes

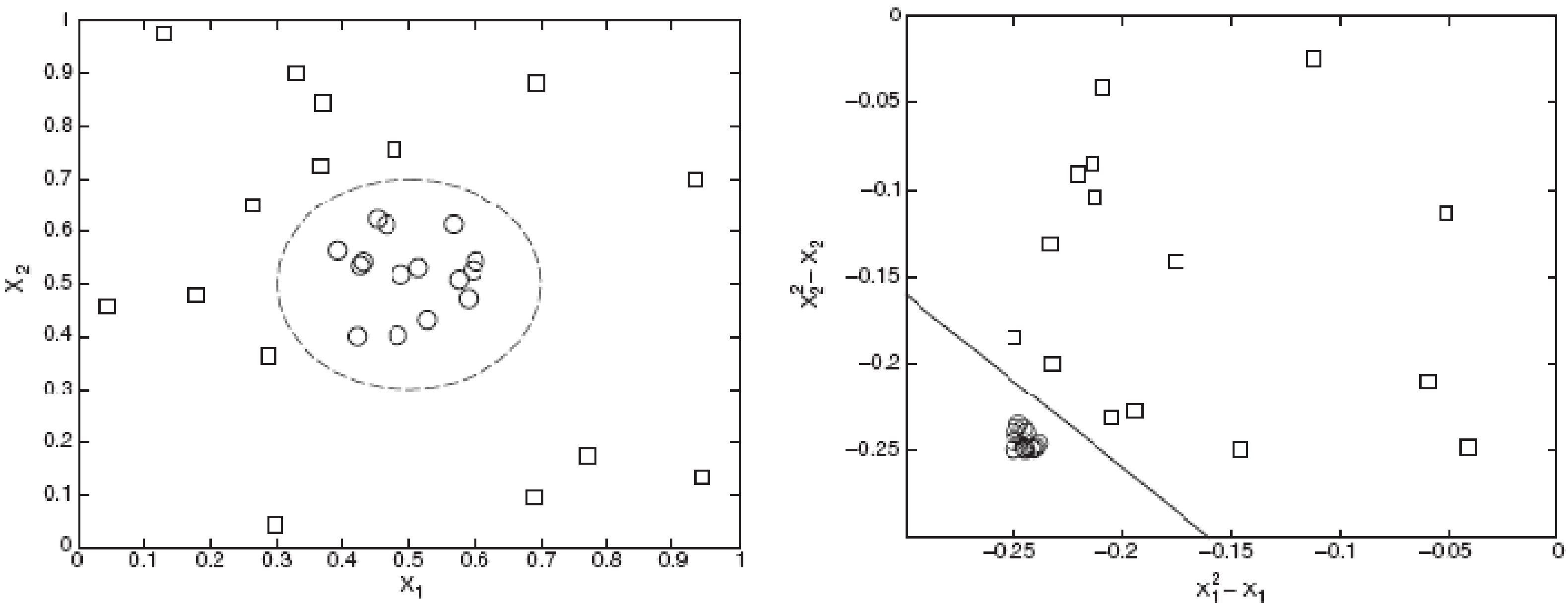
$$\Phi : [x_1, x_2] \rightarrow [x_1, (x_1 + x_2)^4]$$



Support vector machines

Solution: nonlinear transform of attributes

$$\Phi : [x_1, x_2] \rightarrow [(x_1^{\,2} - x_1), (x_2^{\,2} - x_2)]$$



(a) Decision boundary in the original two-dimensional space.

(b) Decision boundary in the transformed space.

Figure 5.28. Classifying data with a nonlinear decision boundary.

Support vector machines

- Issues with finding useful nonlinear transforms
  - Not feasible to do manually as number of attributes grows (i.e. any real world problem)
  - Usually involves transformation to higher dimensional space
    - increases computational burden of SVM optimization
    - curse of dimensionality

- With SVMs, can circumvent all the above via the kernel trick

Support vector machines

- Kernel trick
  - Don't need to specify the attribute transform $\Phi(\mathbf{x})$
  - Only need to know how to calculate the dot product of any two transformed samples:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1) \cdot \Phi(\mathbf{x}_2)$$

Support vector machines

- Kernel trick (cont'd.)
  - The kernel function $\mathrm{k}(\cdot, \cdot)$ is substituted into the dual of the Lagrangian, allowing determination of a maximum margin hyperplane in the (implicitly) transformed space $\Phi(\mathbf{x})$:

$$L_{dual} = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) =$$

$$\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \, \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)$$

  - All subsequent calculations, including predictions on test samples, are done using the kernel in place of $\Phi(\mathbf{x}_1) \cdot \Phi(\mathbf{x}_2)$

Support vector machines

- Common kernel functions for SVM

  – linear $$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$$

  – polynomial $$k(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$$

  – Gaussian or radial basis $$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|^2\right)$$

  – sigmoid $$k(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)$$

Support vector machines

- For some kernels (e.g. Gaussian) the implicit transform $\Phi(\mathbf{x})$ is infinite-dimensional!
  - But calculations with kernel are done in original space, so computational burden and curse of dimensionality aren't a problem.

Support vector machines

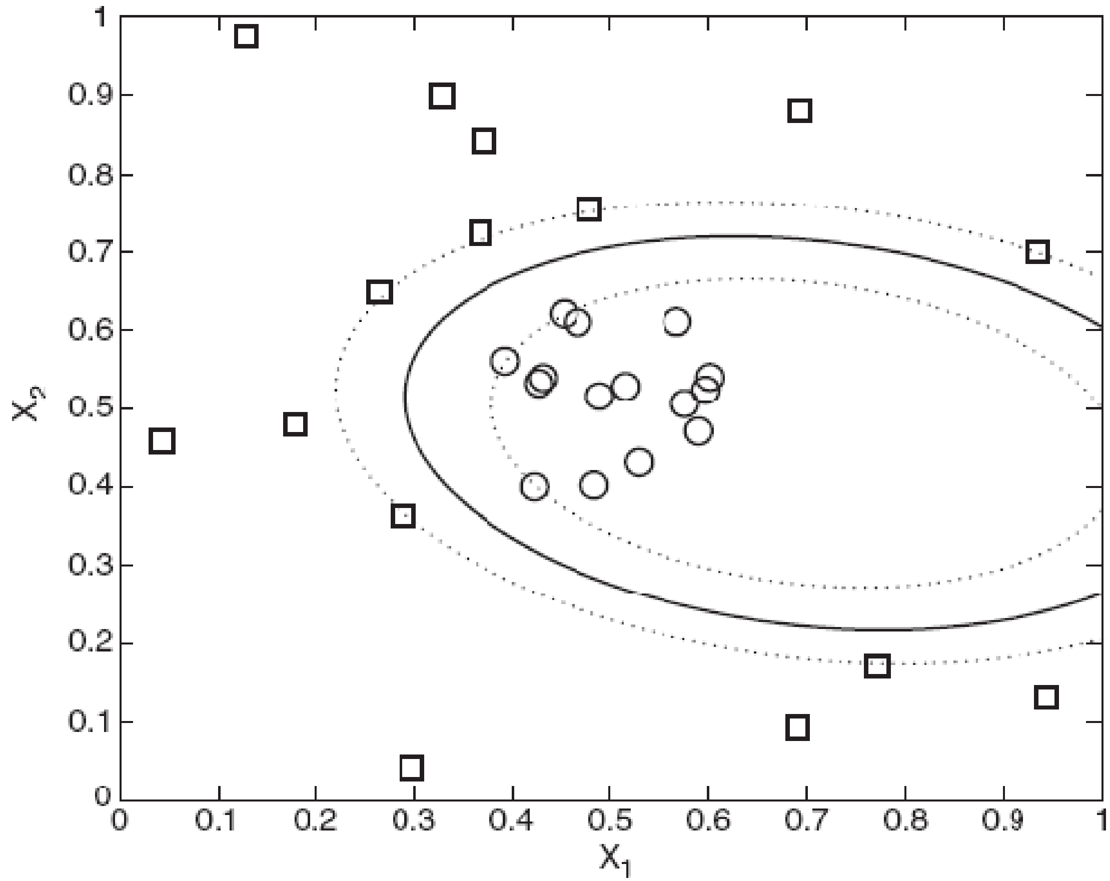


Figure 5.29. Decision boundary produced by a nonlinear SVM with polynomial kernel.

Support vector machines

- Applications of SVMs to machine learning
  - Classification
    - binary
    - multiclass
    - one-class
  - Regression
  - Transduction (semi-supervised learning)
  - Ranking
  - Clustering
  - Structured labels

Support vector machines

- Software
  - SVM*light*
    - http://svmlight.joachims.org/
  - libSVM
    - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
    - includes MATLAB / Octave interface
  - MATLAB svmtrain / svmclassify
    - only supports binary classification

Support vector machines

- Online demos
  - http://cs.stanford.edu/people/karpathy/svmjs/demo/