Support Vector Machines

Ioannis Tsamardinos
Machine Learning Course
Computer Science Department
University of Crete

Support Vector Machines

- Decision surface is a hyperplane (line in 2D) in **feature** space (similar to the Perceptron)
- Arguably, the most important recent discovery in machine learning
- In a nutshell:
  - map the data to a predetermined very high-dimensional space via a kernel function
  - Find the hyperplane that maximizes the margin between the two classes
  - If data are not separable find the hyperplane that maximizes the margin and minimizes the (a weighted average of the) misclassifications

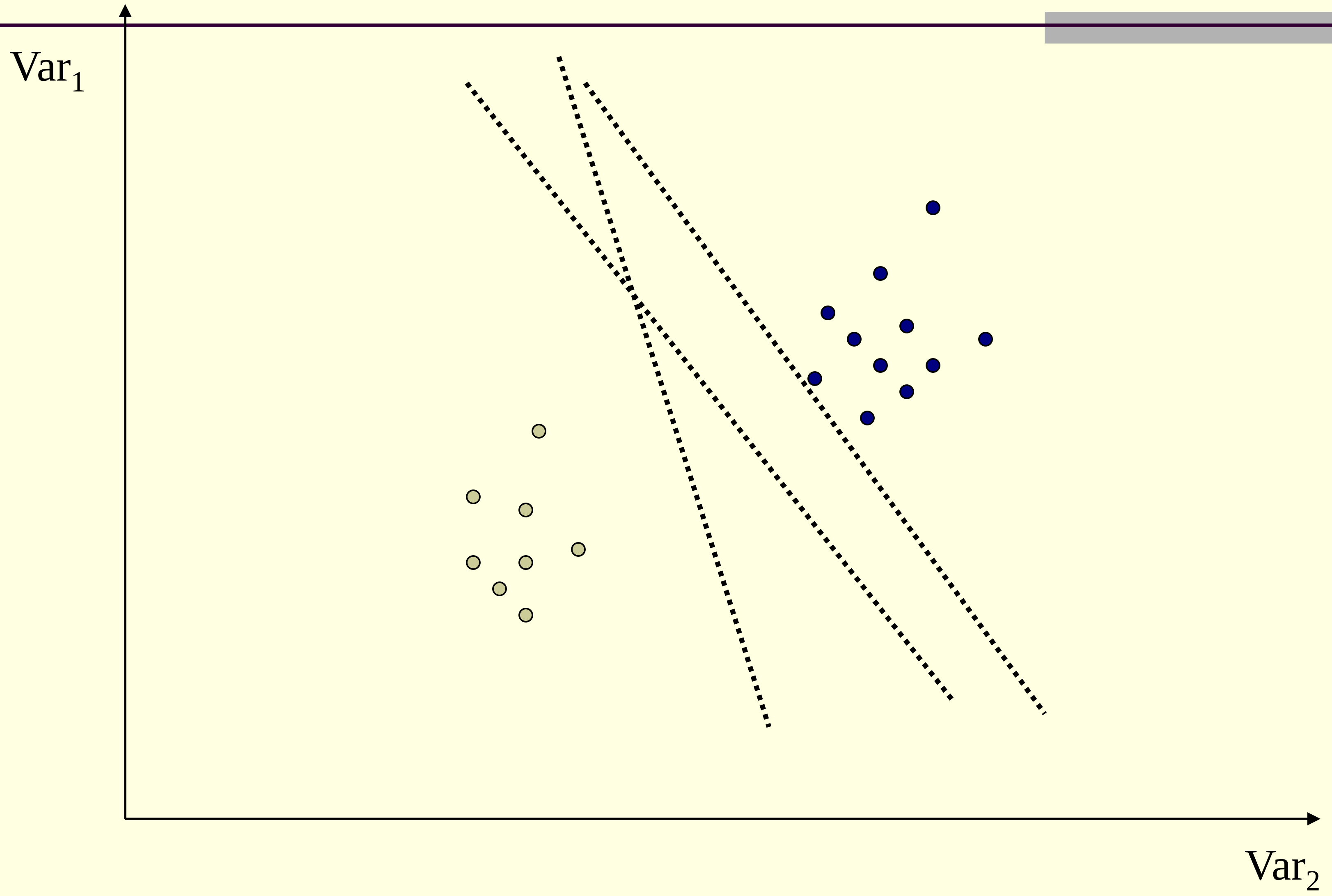
Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (in way that can be identified in a computationally efficient way): maximize margin
  2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
  3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space

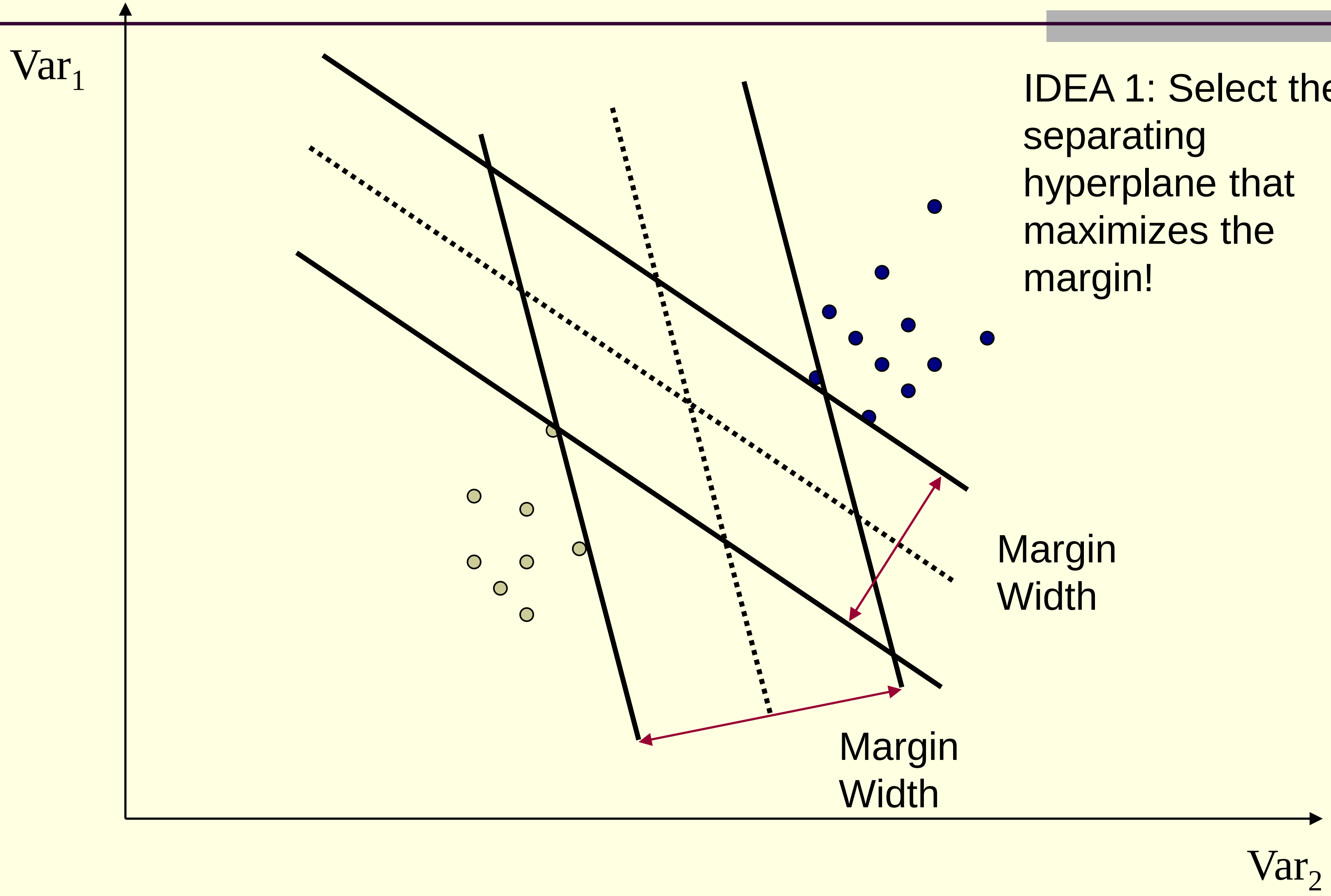
Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (in way that can be identified in a computationally efficient way): maximize margin
  2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
  3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space

Which Separating Hyperplane to Use?



Maximizing the Margin



IDEA 1: Select the separating hyperplane that maximizes the margin!

Why Maximize the Margin?

- Intuitively this feels safest.
- It seems to be the most robust to the estimation of the decision boundary.
- LOOCV is easy since the model is immune to removal of any nonsupport-vector datapoints.
- Theory suggests (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
- It works very well empirically.

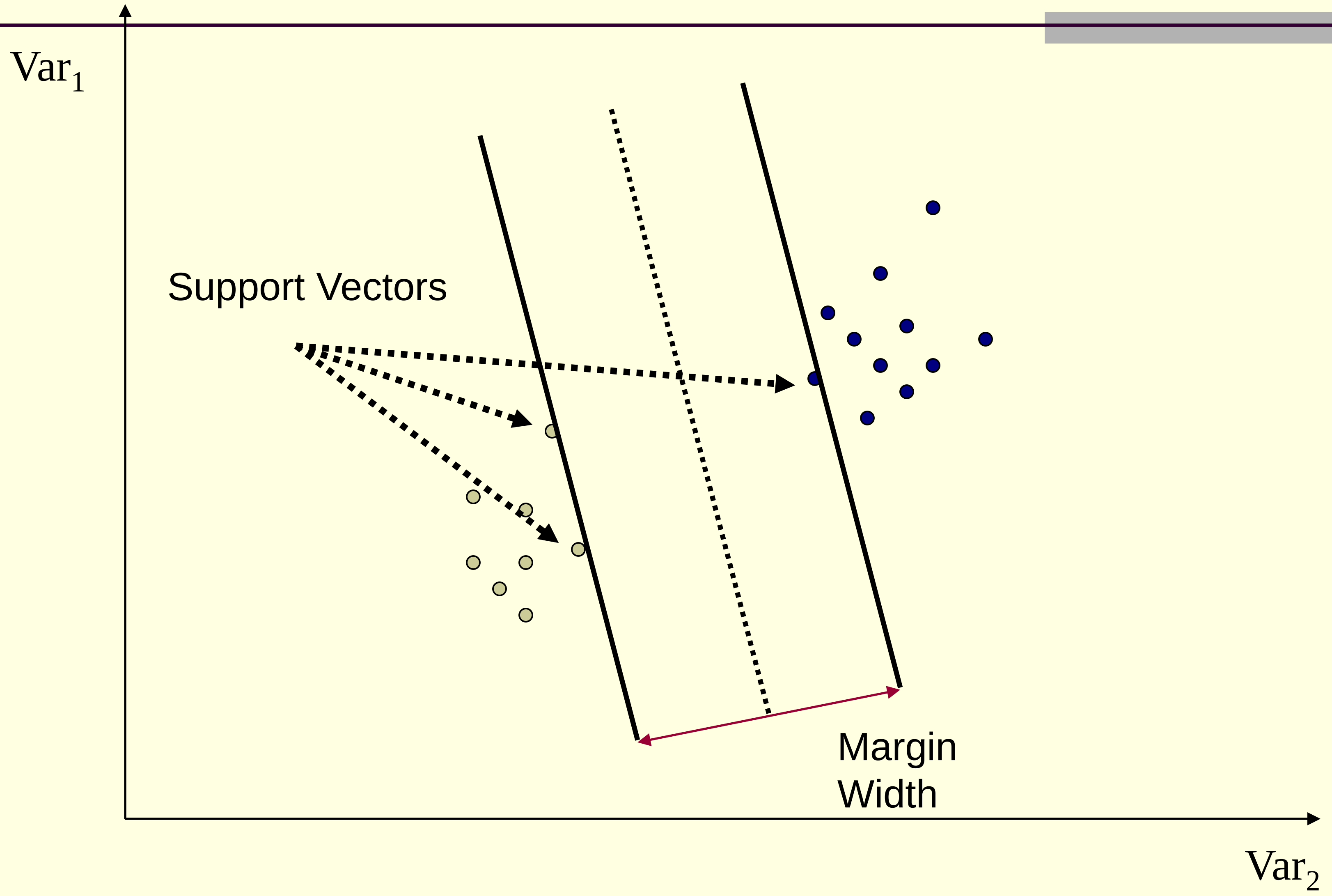
Why Maximize the Margin?

- Perceptron convergence theorem (Novikoff 1962):
  - Let s be the smallest radius of a (hyper)sphere enclosing the data.
  - Suppose there is a w that separates the data, i.e., wx>0 for all x with class 1 and wx<0 for all x with class -1.
  - Let m be the separation margin of the data
  - Let learning rate be 0.5 for the learning rule

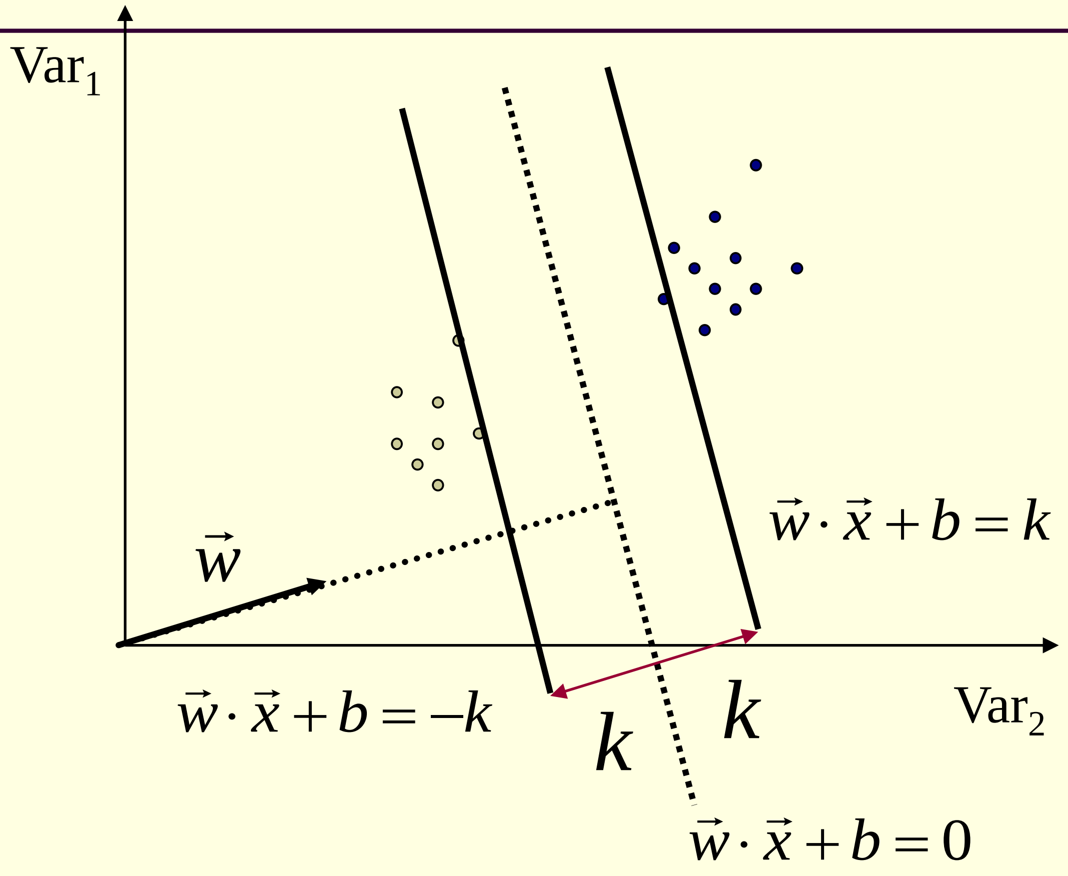
$$\vec{w}' \leftarrow \vec{w} + \eta(t_d - o_d)\vec{x}_d$$

- Then, the number of updates made by the perceptron learning algorithm on the data is at most $(s/m)^2$

Support Vectors



Setting Up the Optimization Problem



The width of the margin is:

$$\frac{2|k|}{\|w\|}$$

So, the problem is:

$$\max \frac{2|k|}{\|w\|}$$

$$s.t.\ (w\cdot x+b)\geq k,\ \forall x \text{ of class 1}$$

$$(w\cdot x+b)\leq -k,\ \forall x \text{ of class 2}$$

Computing the width of the margin

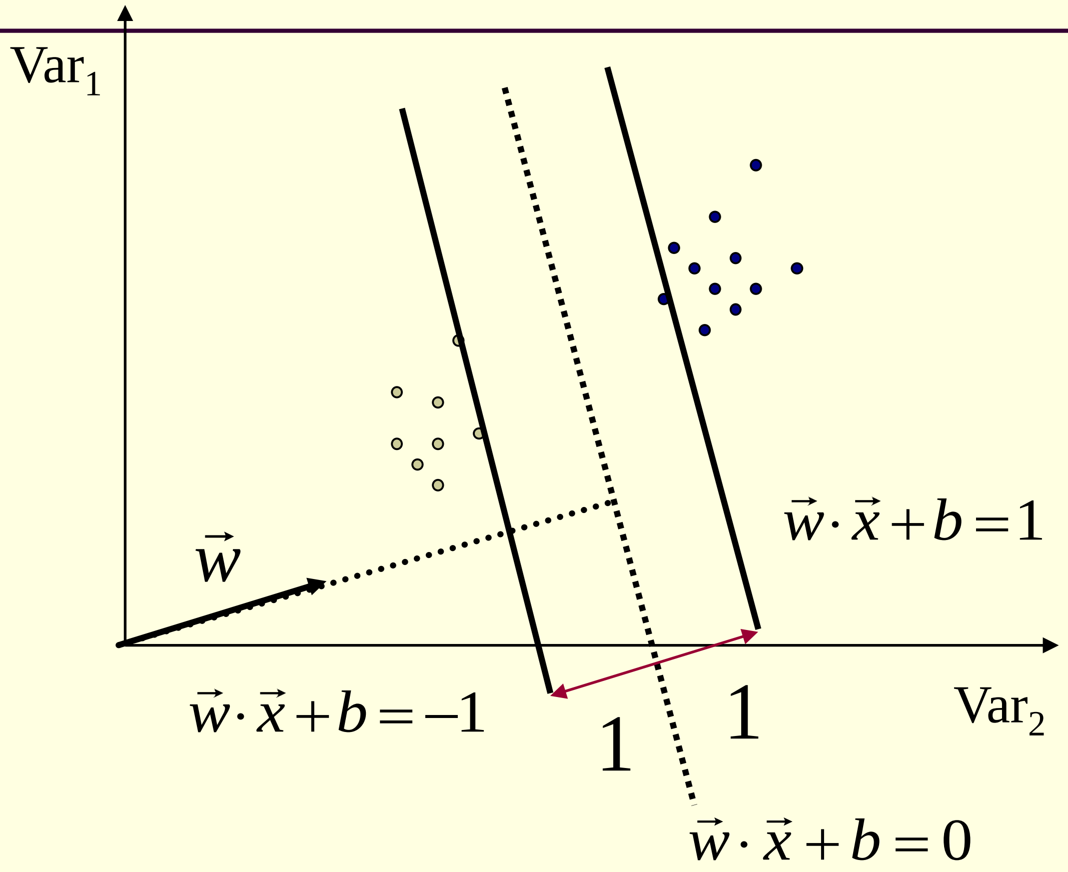
Let x, z points on each hyper-plane of the margin so that they are opposite of each other. Thus, the width is $\left||x-z|\right|$ and (x-z) parallel to w. Then:

$$w \cdot x + b + k = 0,\ w \cdot z + b - k = 0$$

and subtracting the second from the first we get $w \cdot (x-z) + 2k = 0$. We get: $\left||w \cdot (x-z)|\right| = \left||w|\right| \cdot ||(x-z)|| \cdot \cos\theta = |-2k|$. Since (x-z) parallel to w it holds that $cos\theta = 1$ and thus:

$$width = \left||x-z|\right| = \frac{|2k|}{||w||}$$

Setting Up the Optimization Problem



There is a scale and unit for data so that *k=1*. Then problem becomes:

$$\max \frac{2}{\|w\|}$$

$$s.t.\ (w\cdot x+b)\geq 1,\ \forall x \text{ of class } 1$$

$$(w\cdot x+b)\leq -1,\ \forall x \text{ of class } 2$$

Setting Up the Optimization Problem

- If class 1 corresponds to 1 and class 2 corresponds to -1, we can rewrite

$$(w \cdot x_i + b) \geq 1, \ \forall x_i \text{ with } y_i = 1$$
$$(w \cdot x_i + b) \leq -1, \ \forall x_i \text{ with } y_i = -1$$

- as

$$y_i (w \cdot x_i + b) \geq 1, \ \forall x_i$$

- So the problem becomes:

$$\max \frac{2}{\|w\|}$$
$$s.t. \ y_i (w \cdot x_i + b) \geq 1, \ \forall x_i$$

or

$$\min \frac{1}{2} \|w\|^2$$
$$s.t. \ y_i (w \cdot x_i + b) \geq 1, \ \forall x_i$$

Linear, Hard-Margin SVM Formulation

- Find *w,b* that solves

$$\min \frac{1}{2}\|w\|^2$$

$$s.t.\ y_i(w \cdot x_i + b) \geq 1,\ \forall x_i$$

- Problem is convex so, there is a unique global minimum value (when feasible)
- There is also a unique minimizer, i.e. weight and *b* value that provides the minimum (not true for soft-margin SVMs, presented next)
- Non-solvable if the data is not linearly separable

Solving Linear, Hard-Margin SVM

- Quadratic Programming
  - QP is a well-studied class of optimization algorithms to maximize a quadratic function of some real-valued variables subject to linear constraints.
  - Very efficient computationally with modern constraint optimization engines (handles thousands of constraints and training instances).

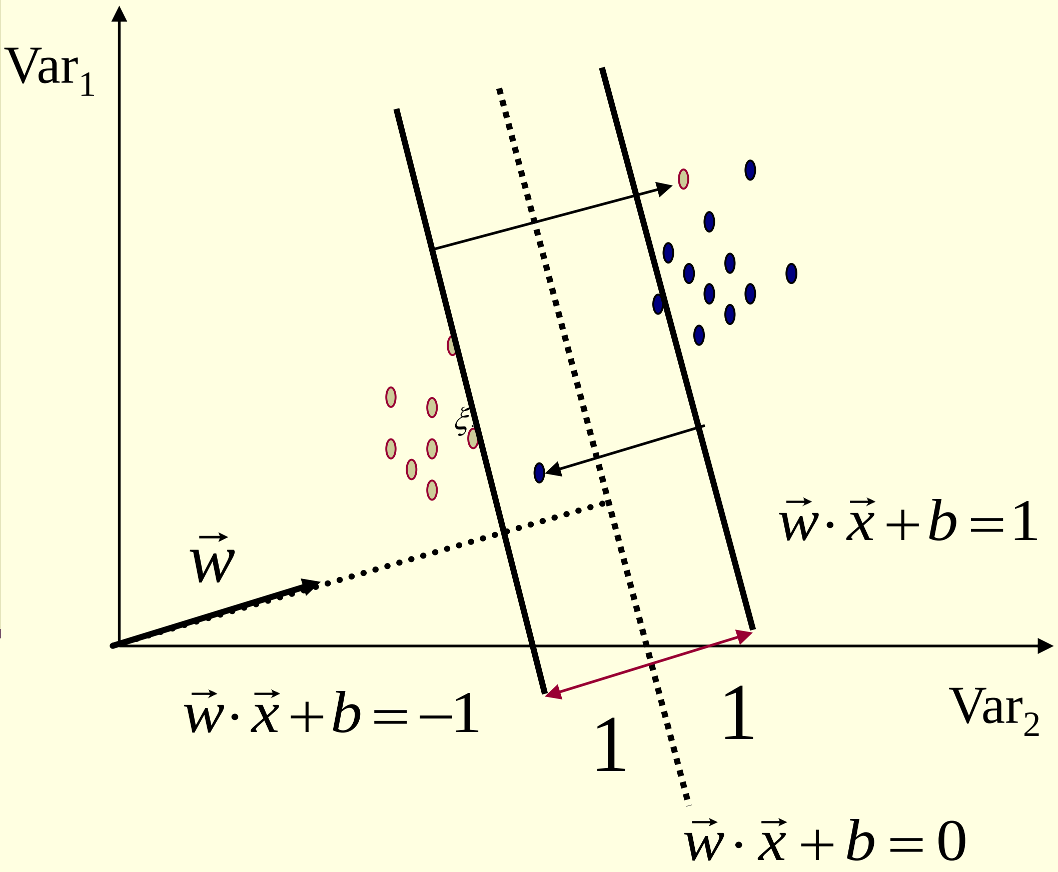
Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (in way that can be identified in a computationally efficient way): maximize margin
  2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
  3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space

Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (in way that can be identified in a computationally efficient way): maximize margin
  2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
  3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space

Non-Linearly Separable Data

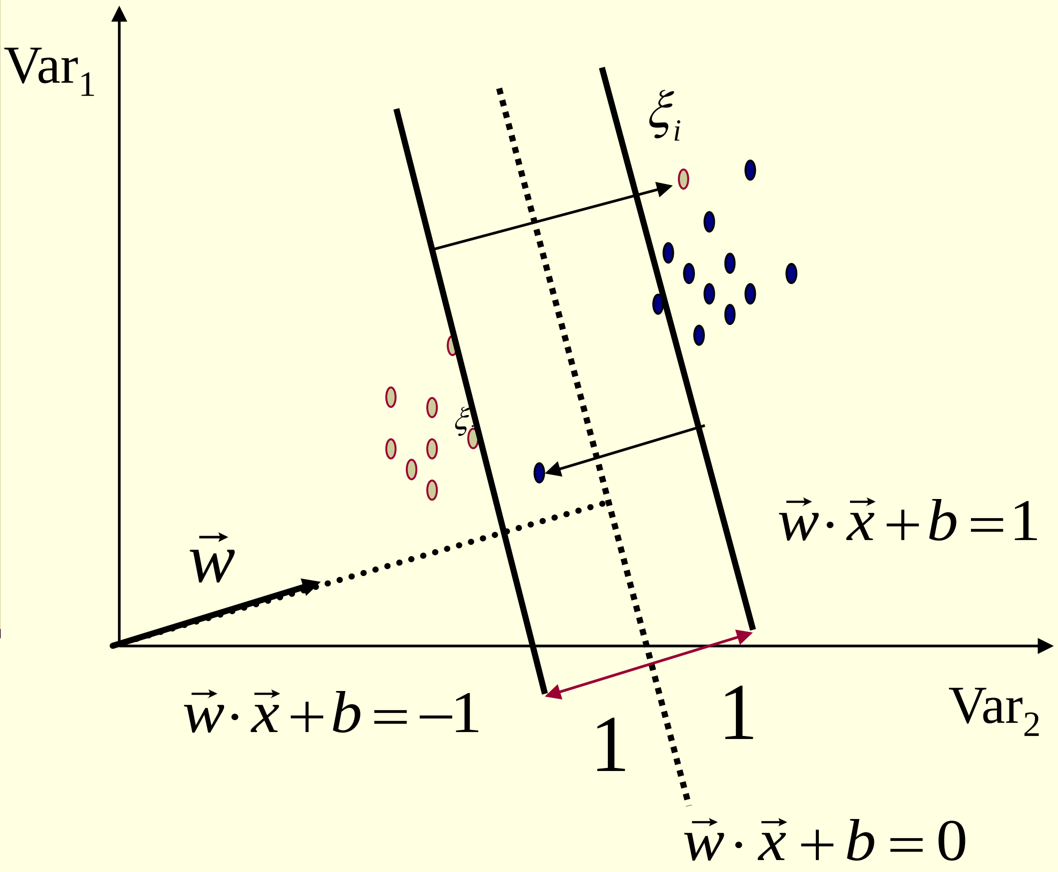


Find hyperplane that minimize both ||w|| **and** the number of misclassifications: ||w||+C*#errors

Problem: NP-complete

Plus, all errors are treated the same

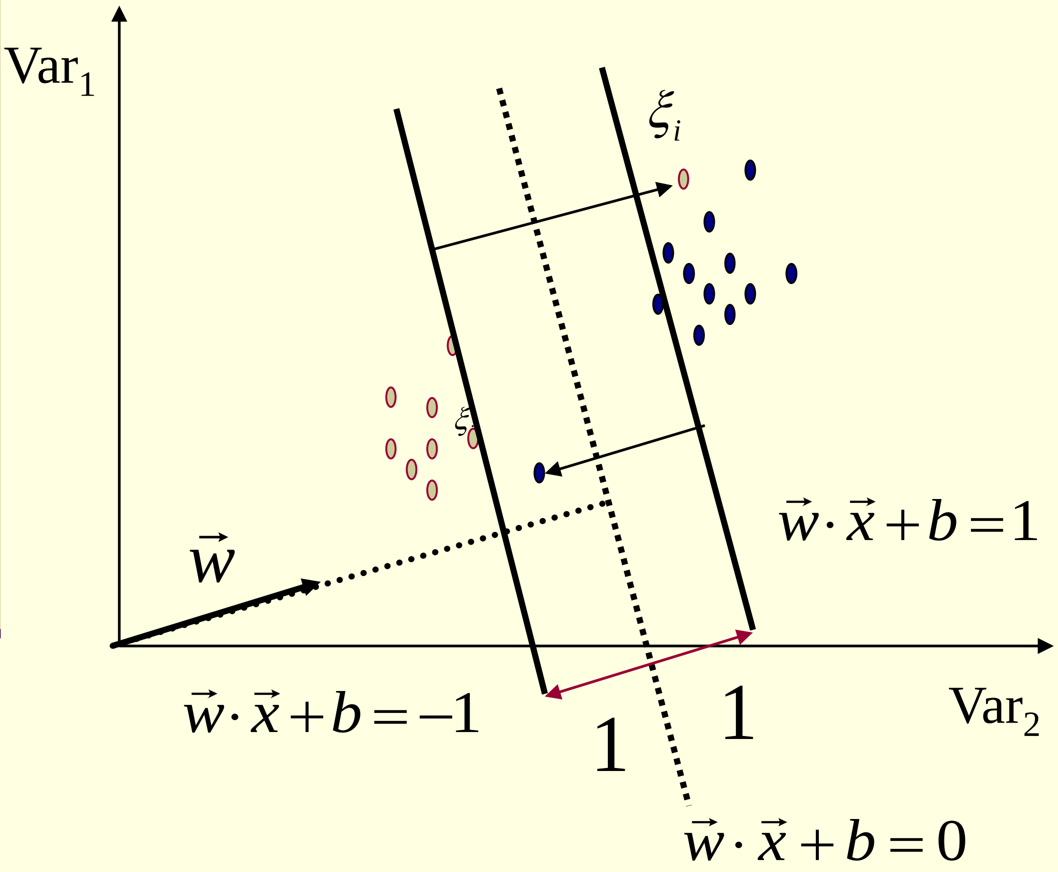
Non-Linearly Separable Data



Minimize ||w||+C*{distance of error points from their desired place}

Allow some instances to fall within the margin, but penalize them

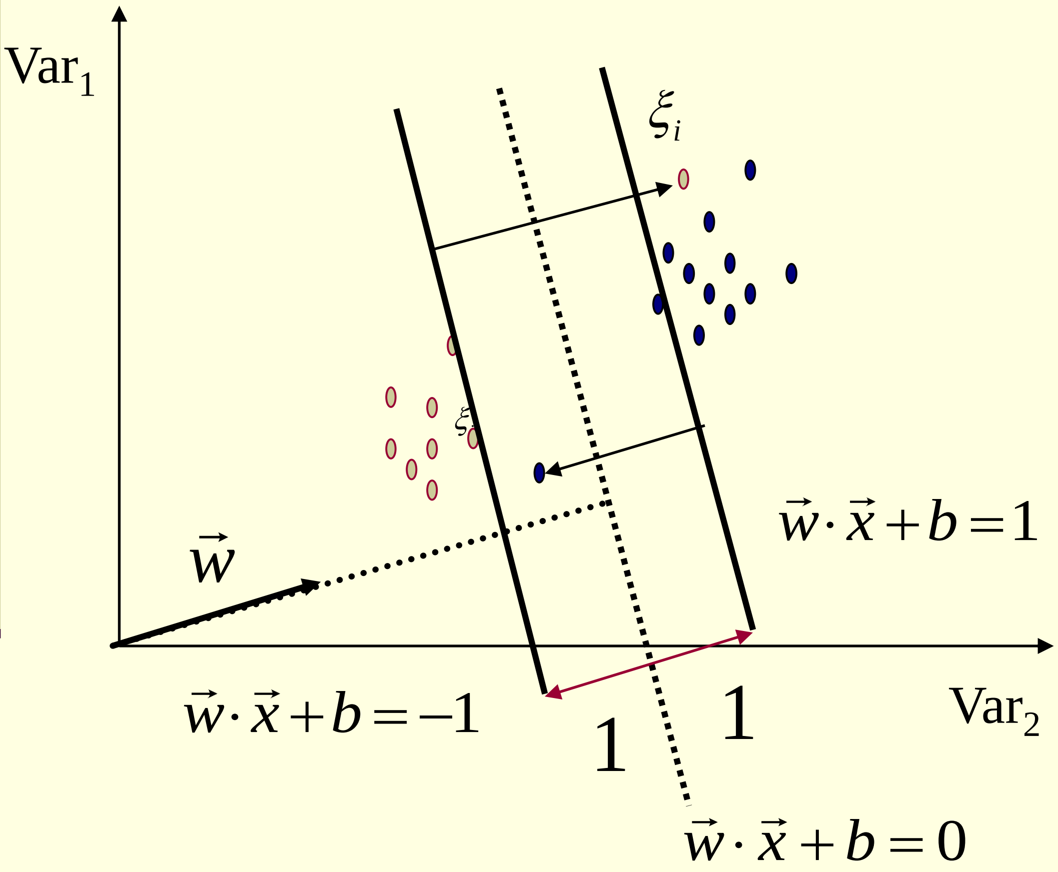
Non-Linearly Separable Data



Introduce slack variables ξ_i

Allow some instances to fall within the margin, but penalize them

Formulating the Optimization Problem



Constraints becomes :

$$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$$
$$\xi_i \geq 0$$

Objective function penalizes for misclassified instances and those within the margin

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

C trades-off margin width and misclassifications

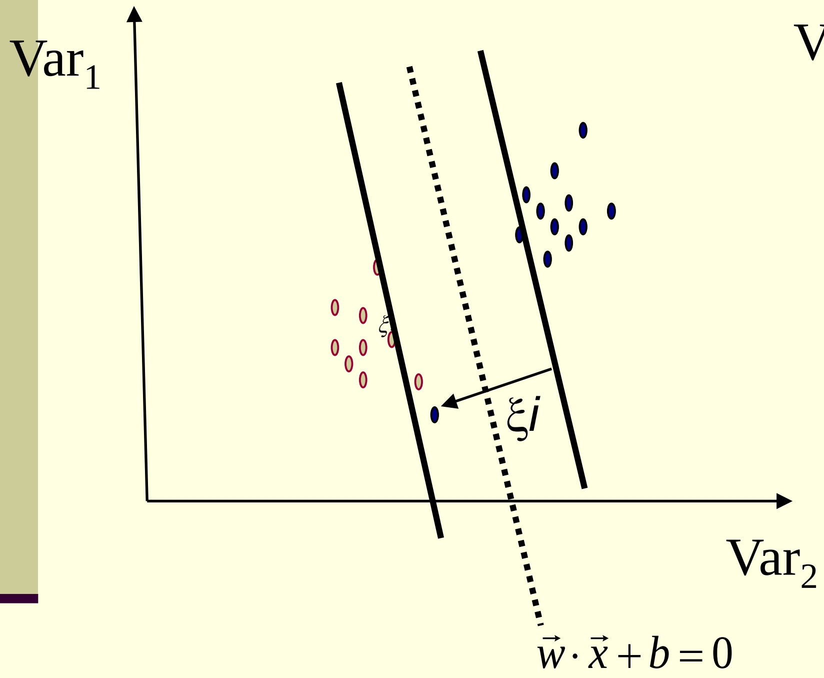
Linear, Soft-Margin SVMs

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

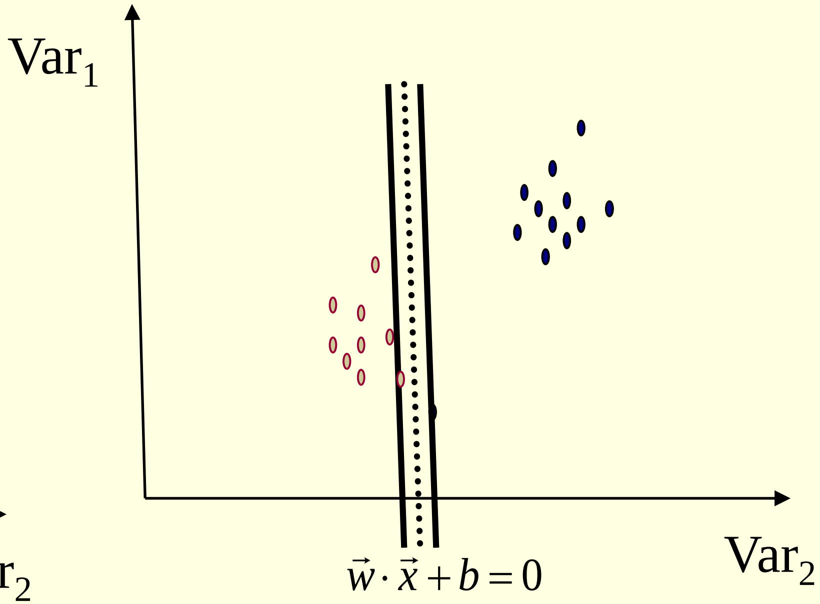
$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \ \forall x_i$$
$$\xi_i \geq 0$$

- Algorithm tries to maintain ξ_i to zero while maximizing margin
- Notice: algorithm does not minimize the *number* of misclassifications (NP-complete problem) but the sum of distances from the margin hyperplanes
- Other formulations use ξ_i^2 instead
- As $C \to \infty$, we get closer to the hard-margin solution
- Hard-margin decision variables = m+1, #constraints = n
- Soft-margin decision variables = m+1+n, #constraints=2n

Robustness of Soft vs Hard Margin SVMs



Soft Margin SVN



Hard Margin SVN

Soft vs Hard Margin SVM

- Soft-Margin always have a solution
- Soft-Margin is more robust to outliers
  - Smoother surfaces (in the non-linear case)
- Hard-Margin does not require to guess the cost parameter (requires no parameters at all)

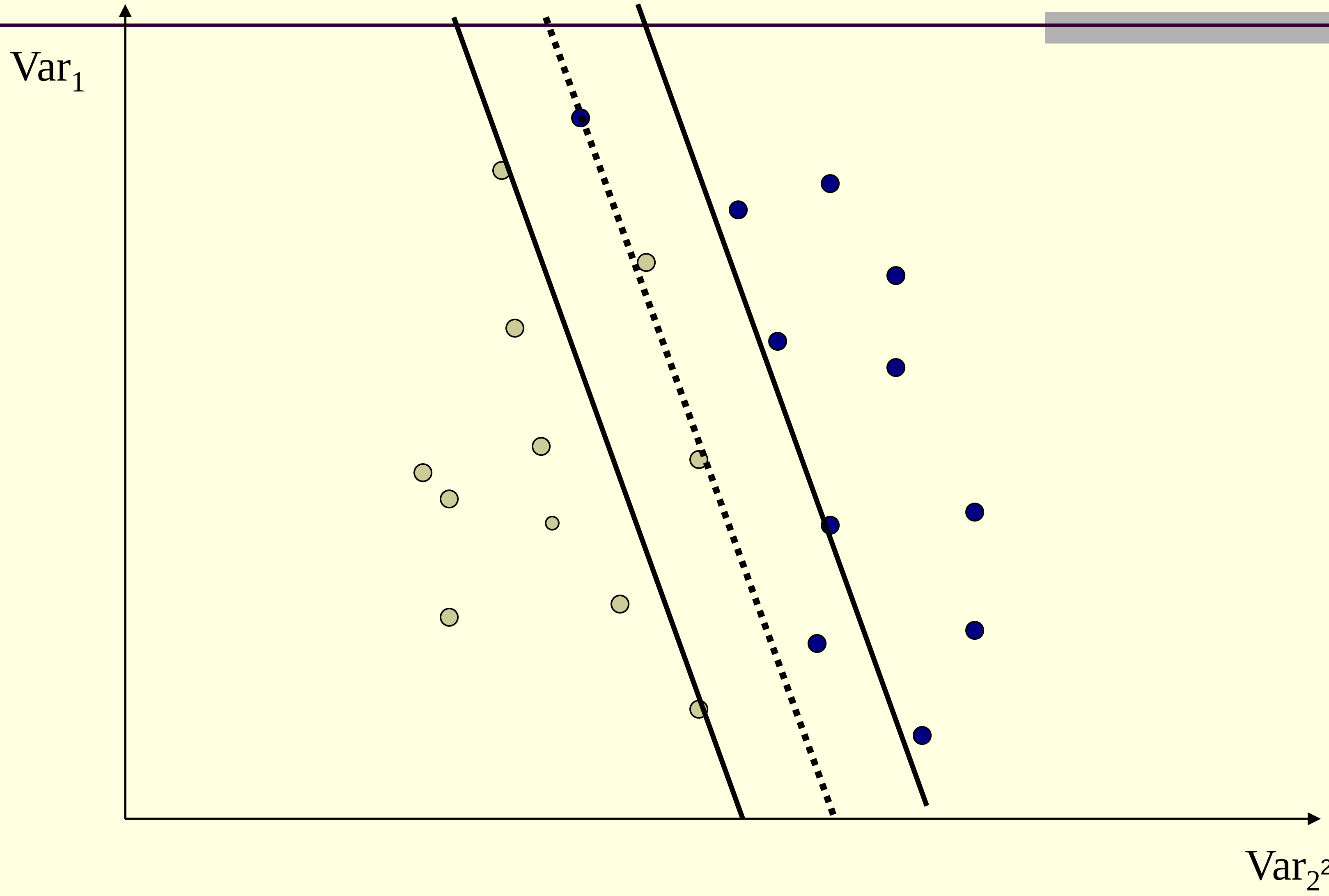
Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (in way that can be identified in a computationally efficient way): maximize margin
  2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
  3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space

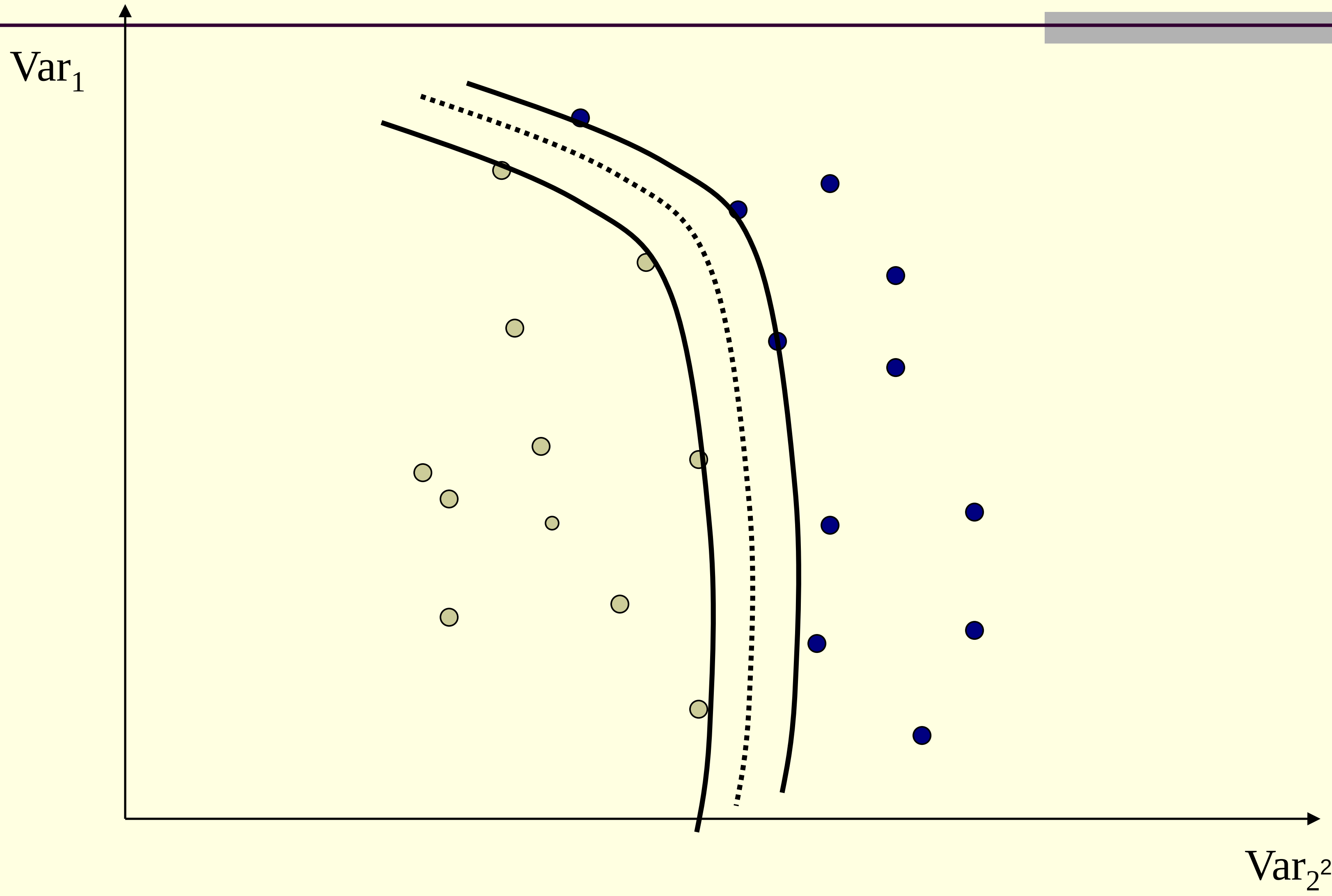
Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (in way that can be identified in a computationally efficient way): maximize margin
  2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
  3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space

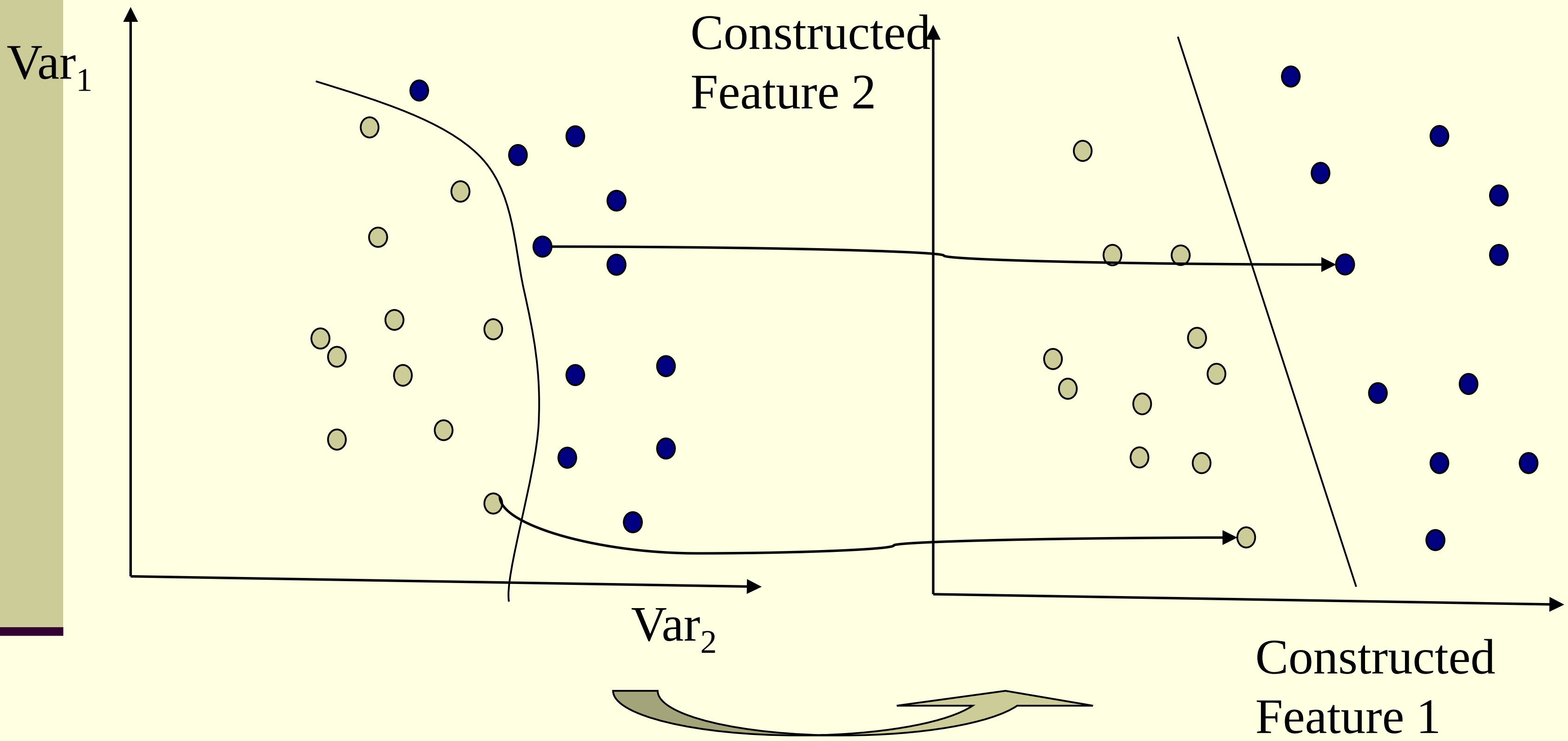
Disadvantages of Linear Decision Surfaces



Advantages of Non-Linear Surfaces



Linear Classifiers in High-Dimensional Spaces



Find function Φ(x) to map to a different space

Mapping Data to a High-Dimensional Space

- Find function $\Phi(x)$ to map to a different space, then SVM formulation becomes:

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \qquad s.t.\ \ y_i(w \cdot \Phi(x) + b) \geq 1 - \xi_i, \forall x_i$$
$$\xi_i \geq 0$$

- Data appear as $\Phi(x)$, weights *w* are now weights in the new space
- Explicit mapping expensive if $\Phi(x)$ is very high dimensional
- Solving the problem without explicitly mapping the data is desirable

The Dual of the SVM Formulation

- Original SVM formulation
  - *n* inequality constraints
  - *n positivity constraints*
  - *n* number of ξ variables

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$s.t. \;\; y_i(w \cdot \Phi(x) + b) \geq 1 - \xi_i, \forall x_i$$

$$\xi_i \geq 0$$

- The (Wolfe) dual of this problem
  - one equality constraint
  - *n* positivity constraints
  - *n* number of α variables (Lagrange multipliers)
  - Objective function more complicated

$$\min_{a_i} \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (\Phi(x_i) \cdot \Phi(x_j)) - \sum_i \alpha_i$$

$$s.t. \;\; \mathrm{C} \geq \alpha_{\mathrm{i}} \geq 0, \forall x_i$$

$$\sum_i \alpha_i y_i = 0$$

- NOTICE: Data only appear as $\Phi(x_i) \cdot \Phi(x_j)$

Computing the Dual

- $L(w, b, \xi, \alpha, r)$
$= \frac{1}{2} w \cdot w + C \sum \xi_i + \sum \alpha_i [1 - \xi_i - \mathrm{y_i}(\mathrm{w} \cdot x_i + b)] - \sum r_i \xi_i$

$$\frac{\partial L}{\partial w} = w - \sum a_i y_i x_i = 0 \Rightarrow w = \sum a_i y_i x_i \ (1)$$

Thus, the weight vector is a linear combination of the support vectors!

$$\frac{\partial L}{\partial b} = -\sum a_i y_i = 0 \Rightarrow \sum a_i y_i = 0 \ (2)$$

$\frac{\partial L}{\partial \xi_k} = C - a_i - r_i = 0 \Rightarrow a_i \leq C$ (3) because r_i are non-negative.

Computing the Dual

- $\max L, s.t\ a_i \geq 0, r_i \geq 0$ and constraints (1), (2), (3)

Substitute (1)-(3) to L to get the result

The Kernel Trick

- $\Phi(x_i) \cdot \Phi(x_j)$: means, map data into new space, then take the inner product of the new vectors
- We can find a function such that: $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$ easily computable
- Then, we do not need to explicitly map the data into the high-dimensional space to solve the optimization problem (for training)
- How do we classify without explicitly mapping the new instances? Turns out

$$\operatorname{sgn}(wx + b) = \operatorname{sgn}\left(\sum_i \alpha_i y_i K(x_i, x) + b\right)$$

$$\text{where } b \text{ solves } \alpha_j \left(y_j \sum_i \alpha_i y_i K(x_i, x_j) + b - 1\right) = 0,$$

$$\text{for any } j \text{ with } 0 < \alpha_j < C$$

Examples of Kernels

- Assume we measure two quantities, e.g. expression level of genes *TrkC* and *SonicHedghog (SH)* and we use the mapping:

$$\Phi :< x_{TrkC}, x_{SH} > \rightarrow \{x^2_{TrkC}, x^2_{SH}, \sqrt{2}x_{TrkC}x_{SH}, x_{TrkC}, x_{SH}, 1\}$$

- Consider the function:

$$K(x \cdot z) = (x \cdot z + 1)^2$$

- We can verify that:

$$\Phi(x) \cdot \Phi(z) =$$

$$x^2_{TrkC} z^2_{TrkC} + x^2_{SH} z^2_{SH} + 2x_{TrkC}x_{SH}z_{TrkC}z_{SH} + x_{TrkC}z_{TrkC} + x_{SH}z_{SH} + 1 =$$

$$= (x_{TrkC}z_{TrkC} + x_{SH}z_{SH} + 1)^2 = (x \cdot z + 1)^2 = K(x, z)$$

Polynomial and Gaussian Kernels

$$K(x,z) = (x \cdot z + 1)^p$$

- is called the polynomial kernel of degree *p*.
- For *p*=2, if we measure 7,000 genes using the kernel once means calculating a summation product with 7,000 terms then taking the square of this number
- Mapping explicitly to the high-dimensional space means calculating approximately 50,000,000 new features for both training instances, then taking the inner product of that (another 50,000,000 terms to sum)
- In general, using the Kernel trick provides huge computational savings over explicit mapping!
- Another commonly used Kernel is the Gaussian (maps to a dimensional space with number of dimensions equal to the number of training cases):

$$K(x,z) = \exp(-\|x - z\| / 2\sigma^2)$$

The Mercer Condition

- Is there a mapping $\Phi(x)$ for any symmetric function $K(x,z)$? No
- The SVM dual formulation requires calculation $K(x_i , x_j)$ for each pair of training instances. The array $G_{ij} = K(x_i , x_j)$ is called the Gram matrix
- There is a feature space $\Phi(x)$ when the Kernel is such that G is always semi-positive definite (Mercer condition)

Geometry of SVM model

- Where does a point *x* falls with respect to the margin?

Geometry of SVM model

- Where does a point *x* falls with respect to the margin?
- If $a_i = 0$, the corresponding constraint is inactive, then x_i on the margin (degenerate case) or **strictly in the correct side of the margin**
- If $a_i > 0$, the point is a **support vector,** the corresponding constraint is active and x_i is on the margin or the wrong side of it
  - If $a_i < C$, then $r_i > 0$, then the corresponding constraint is active and thus $\xi_I = 0$: the point is **exactly on the margin**
  - If $a_i = C$, then $r_i = 0$, then the corresponding constraint is inactive and thus $\xi_I > 0$: the point is at the **wrong side of the margin**

Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (in way that can be identified in a computationally efficient way): maximize margin
  2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
  3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space

Other Types of Kernel Methods

- SVMs that perform regression
- SVMs that perform clustering
- ν-Support Vector Machines: maximize margin while bounding the number of margin errors
- Leave One Out Machines: minimize the bound of the leave-one-out error
- SVM formulations that take into consideration difference in cost of misclassification for the different classes
- Kernels suitable for sequences of strings, or other specialized kernels

Variable Selection with SVMs

- Recursive Feature Elimination
  - Train a linear SVM
  - Remove the variables with the lowest weights (those variables affect classification the least), e.g., remove the lowest 50% of variables
  - Retrain the SVM with remaining variables and repeat until classification is reduced
- Very successful
- Other formulations exist where minimizing the number of variables is folded into the optimization problem
- Similar algorithm exist for non-linear SVMs
- Some of the best and most efficient variable selection methods

Comparison with Neural Networks

Neural Networks	SVMs
Hidden Layers map to lower dimensional spaces	Kernel maps to a very-high dimensional space
Search space has multiple local minima	Search space has a unique minimum
Training is expensive	Training is extremely efficient
Classification extremely efficient	Classification extremely efficient
Requires number of hidden units and layers	Kernel and cost the two parameters to select
Very good accuracy in typical domains	Very good accuracy in typical domains
	Extremely robust

Why do SVMs Generalize?

- Even though they map to a very high-dimensional space
  - They have a very strong bias in that space
  - The solution has to be a linear combination of the training instances
- Large theory on Structural Risk Minimization providing bounds on the error of an SVM
  - Typically the error bounds too loose to be of practical use

MultiClass SVMs

- One-versus-all
  - Train *n* binary classifiers, one for each class against all other classes.
  - Predicted class is the class of the most confident classifier
- One-versus-one
  - Train *n(n-1)/2* classifiers, each discriminating between a pair of classes
  - Several strategies for selecting the final classification based on the output of the binary SVMs
- Truly MultiClass SVMs
  - Generalize the SVM formulation to multiple categories
- More on that in the nominated for the student paper award: "Methods for Multi-Category Cancer Diagnosis from Gene Expression Data: A Comprehensive Evaluation to Inform Decision Support System Development", Alexander Statnikov, Constantin F. Aliferis, Ioannis Tsamardinos

Conclusions

- SVMs express learning as a mathematical program taking advantage of the rich theory in optimization
- SVM uses the kernel trick to map indirectly to extremely high dimensional spaces
- SVMs extremely successful, robust, efficient, and versatile while there are good theoretical indications as to why they generalize well

Suggested Further Reading

- http://www.kernel-machines.org/tutorial.html
- C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998.
- P.H. Chen, C.-J. Lin, and B. Schölkopf. A tutorial on nu -support vector machines. 2003.
- N. Cristianini. ICML'01 tutorial, 2001.
- K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Neural Networks*, 12(2):181-201, May 2001. (PDF)
- B. Schölkopf. SVM and kernel methods, 2001. Tutorial given at the NIPS Conference.
- Hastie, Tibshirani, Friedman, The Elements of Statistical Learning, Springel 2001