# Support Vector Machines (Formal : Version 1)

# 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 (penalty associated with) misclassifications

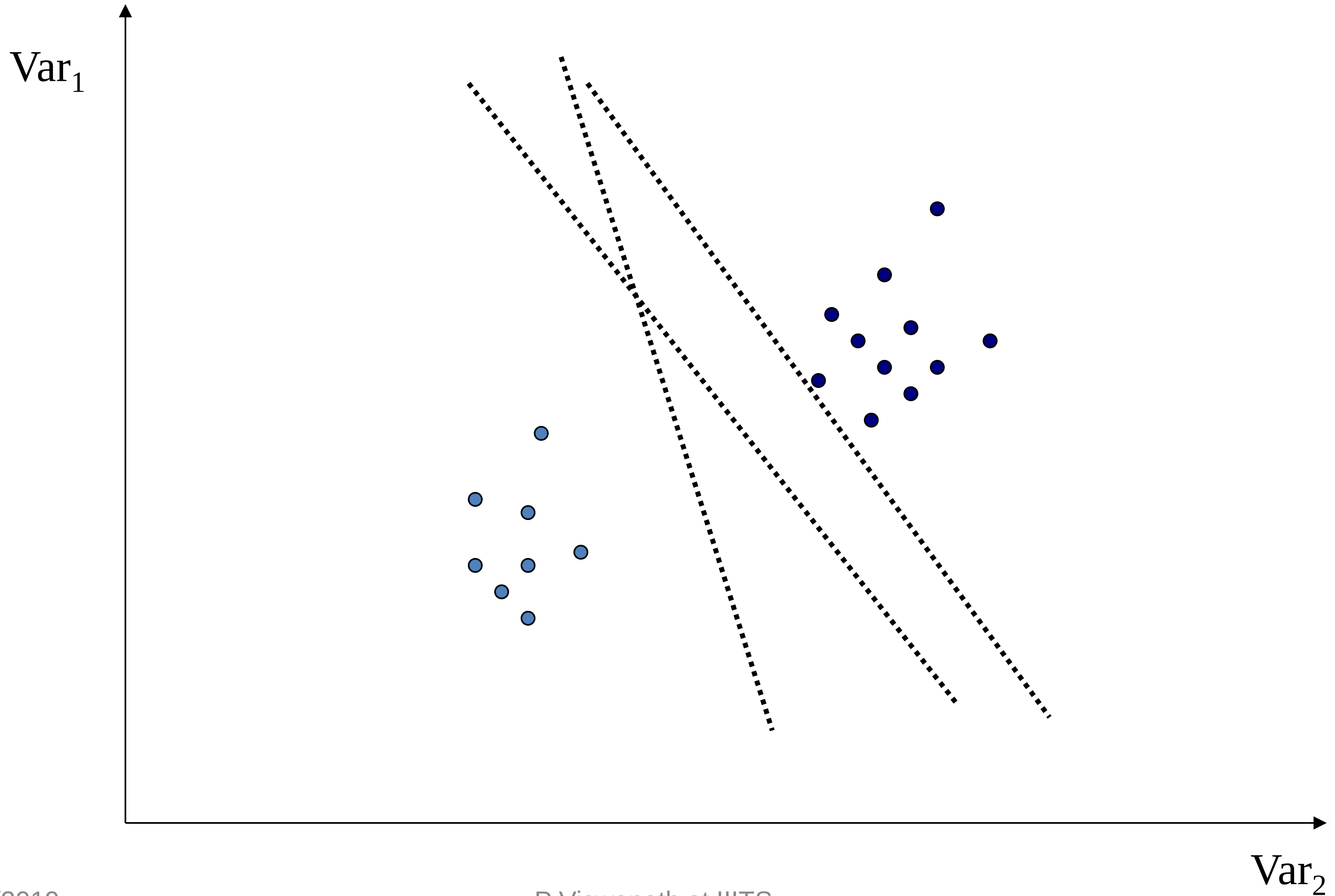
# Support Vector Machines

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

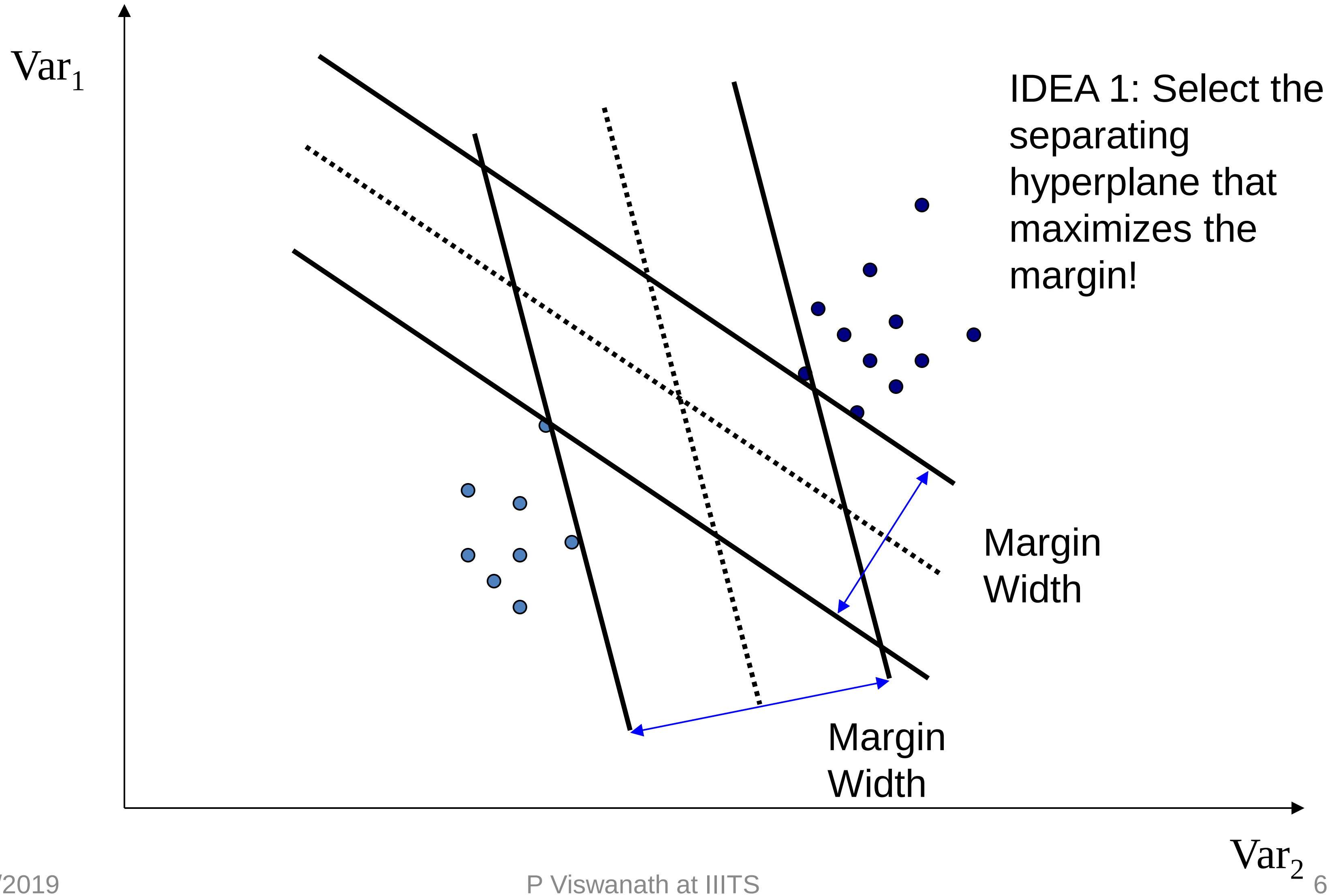
# Support Vector Machines

- 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?

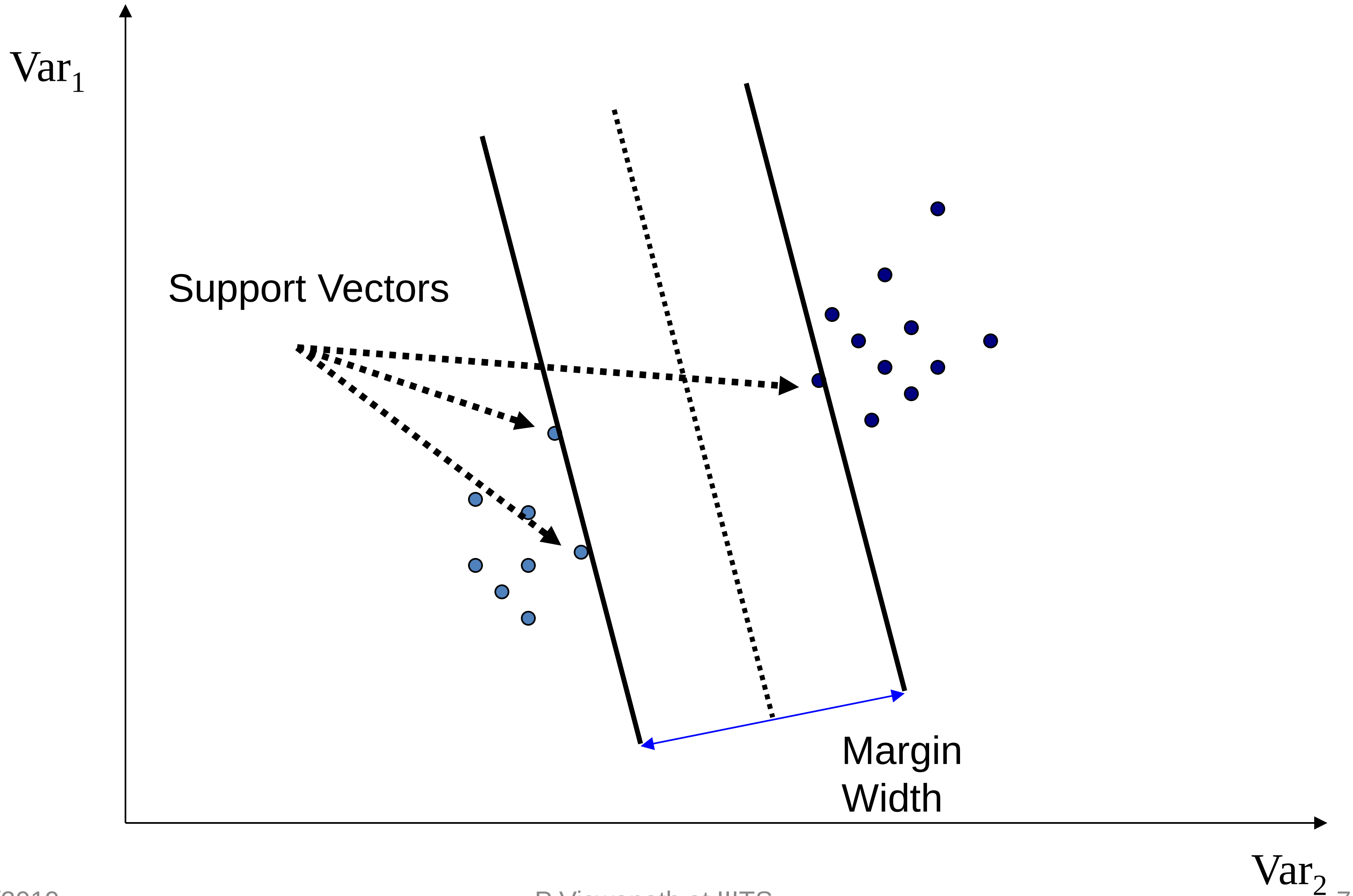


# Maximizing the Margin

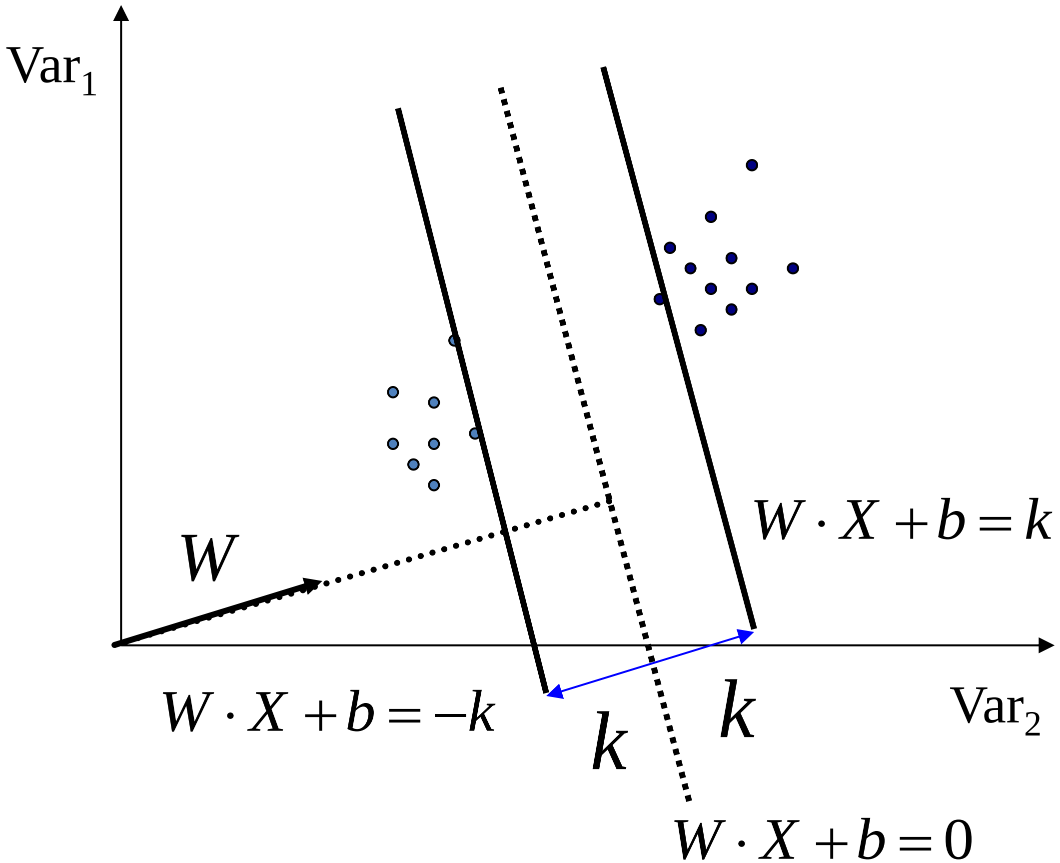



IDEA 1: Select the separating hyperplane that maximizes the margin!

# Support Vectors



# Setting Up the Optimization Problem




The width of the margin is:

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

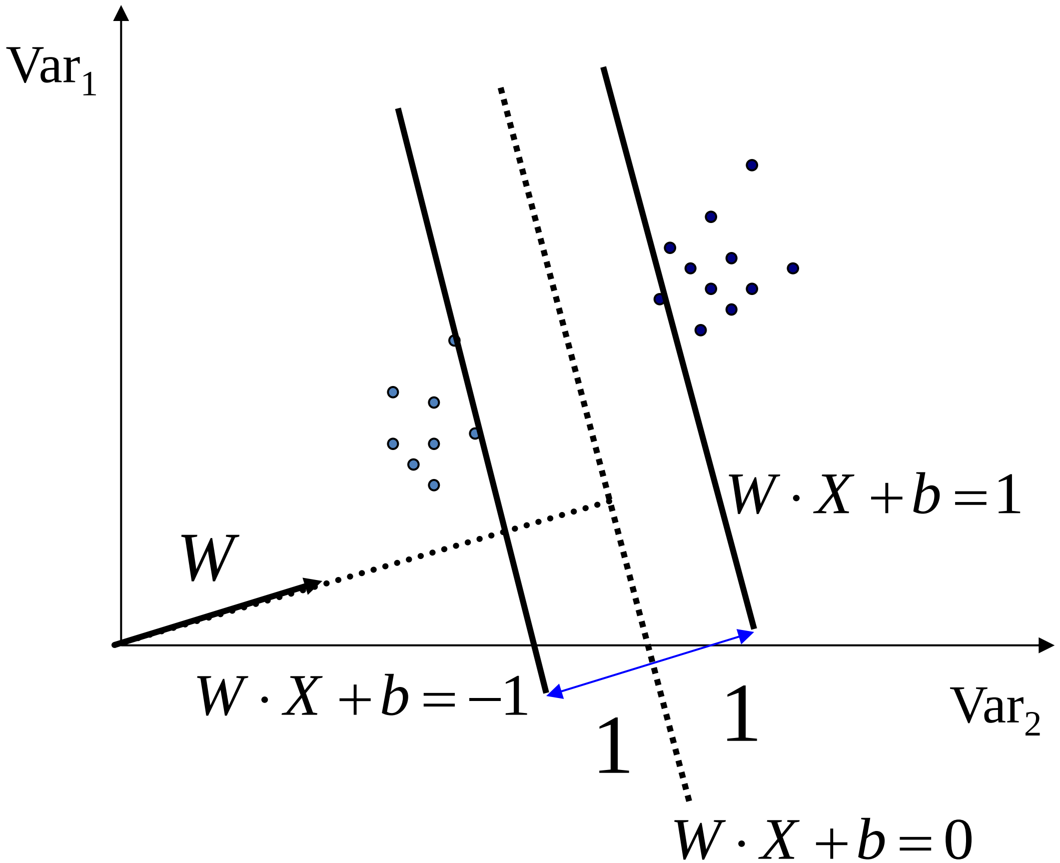
**So, the problem is:**

Maximize $\frac{2|k|}{\|W\|}$

s.t. $W \cdot X + b \geq k, \forall X \text{ in Class } 1$

$W . X + b \leq -k, \forall X \text{ in Class } 2$

# Setting Up the Optimization Problem




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

**So, the problem is:**

Maximize $\frac{2}{\|W\|}$

s.t. $W \cdot X + b \geq 1, \forall\, X \text{ in } Class\ 1$

$W . X + b \leq -1, \forall\, X \text{ in } Class\ 2$

# Setting Up the Optimization Problem

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

$$
\begin{aligned}
W \cdot X_i + b &\geq 1, \forall X_i \; with \; y_i = 1 \\
W . X_i + b &\leq -1, \forall X_i \; with \; y_i = -1
\end{aligned}
$$

- as

$$\mathrm{y_i}(W \cdot X_i + b) \geq 1, \forall X_i$$

- So the problem becomes:

Maximize $\frac{2}{\|W\|}$
s.t. $\mathrm{y_i}(W \cdot X_i + b) \geq 1, \forall X_i$

or

Minimize $\frac{1}{2}\|W\|^2$
s.t. $\mathrm{y_i}(W \cdot X_i + b) \geq 1, \forall X_i$

# Linear, Hard-Margin SVM Formulation

- Find *W, b* that solves

| |
|---|
| Minimize $\frac{1}{2}\|W\|^2$<br>s.t. $\mathrm{y_i}(W \cdot X_i + b) \geq 1, \forall X_i$ |

- Problem is convex so, there is a unique global minimum value (when feasible)
- Non-solvable if the data is not linearly separable
- Quadratic Programming
  - Very efficient computationally with modern constraint optimization engines (handles thousands of constraints).

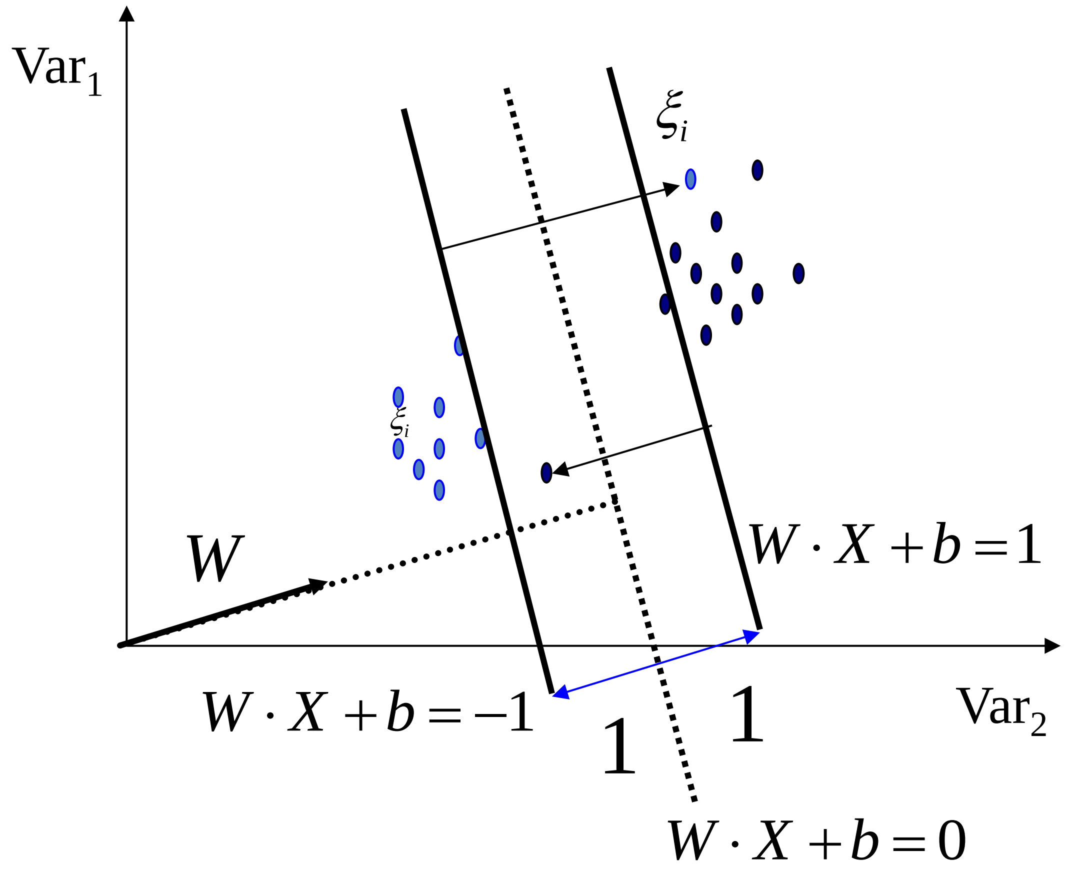
# Support Vector Machines

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

# 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

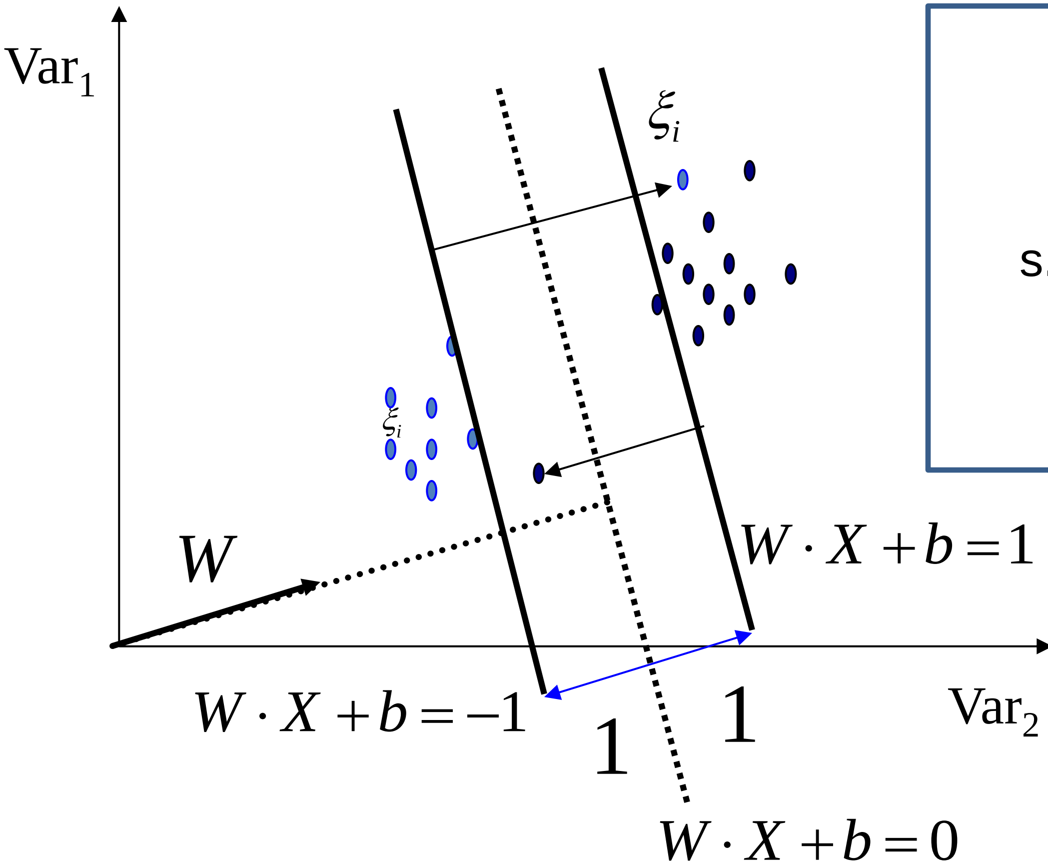
# Non-Linearly Separable Data




Introduce slack variables $\xi_i$

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

# Non-Linearly Separable Data




$$\text{Min } \frac{1}{2}\|W\|^2 + C\sum_i \xi_i$$

s.t. $y_i(W \cdot X_i + b) \geq 1 - \xi_i, \forall X_i$

and $\xi_i \geq 0$

*C* trades-off margin width and misclassifications

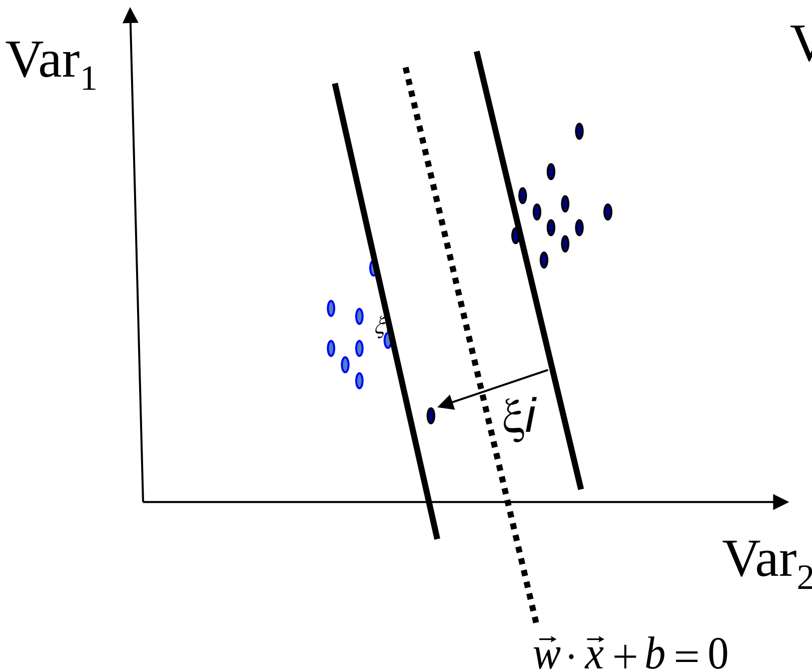
# Linear, Soft-Margin SVMs

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

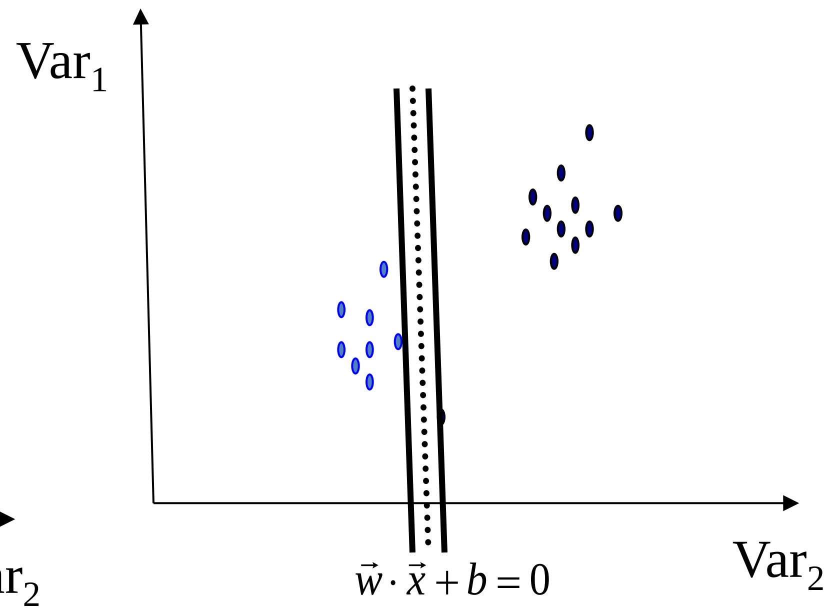
$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \ \forall x_i$$
$$\xi_i \geq 0$$

- Algorithm tries to maintain $\xi_i$ 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 $\xi_i^2$ instead
- As $C \rightarrow \infty$, we get closer to the hard-margin solution

# Robustness of Soft vs Hard Margin SVMs




Soft Margin SVM




Hard Margin SVM

# 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)

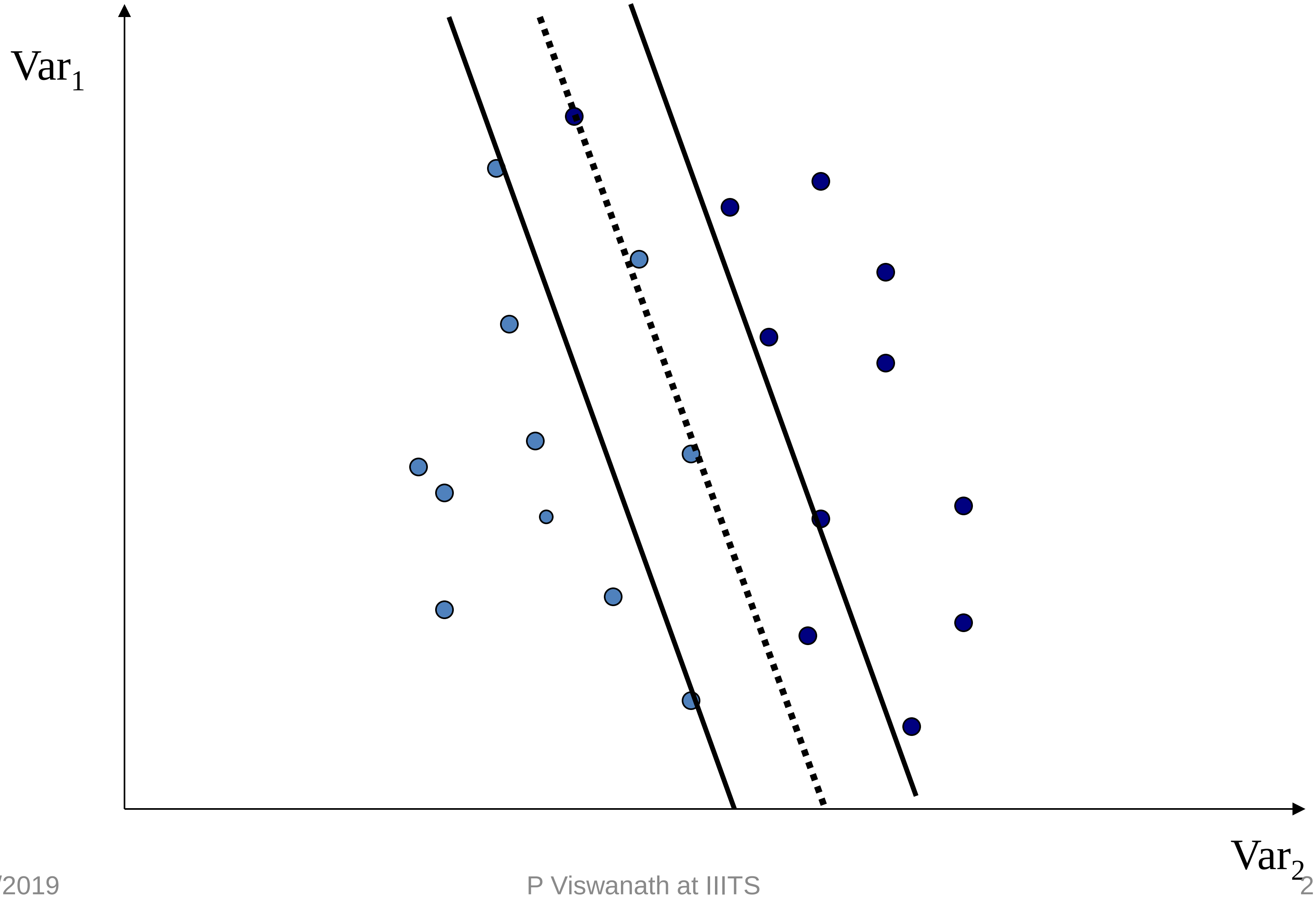
# Support Vector Machines

- 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

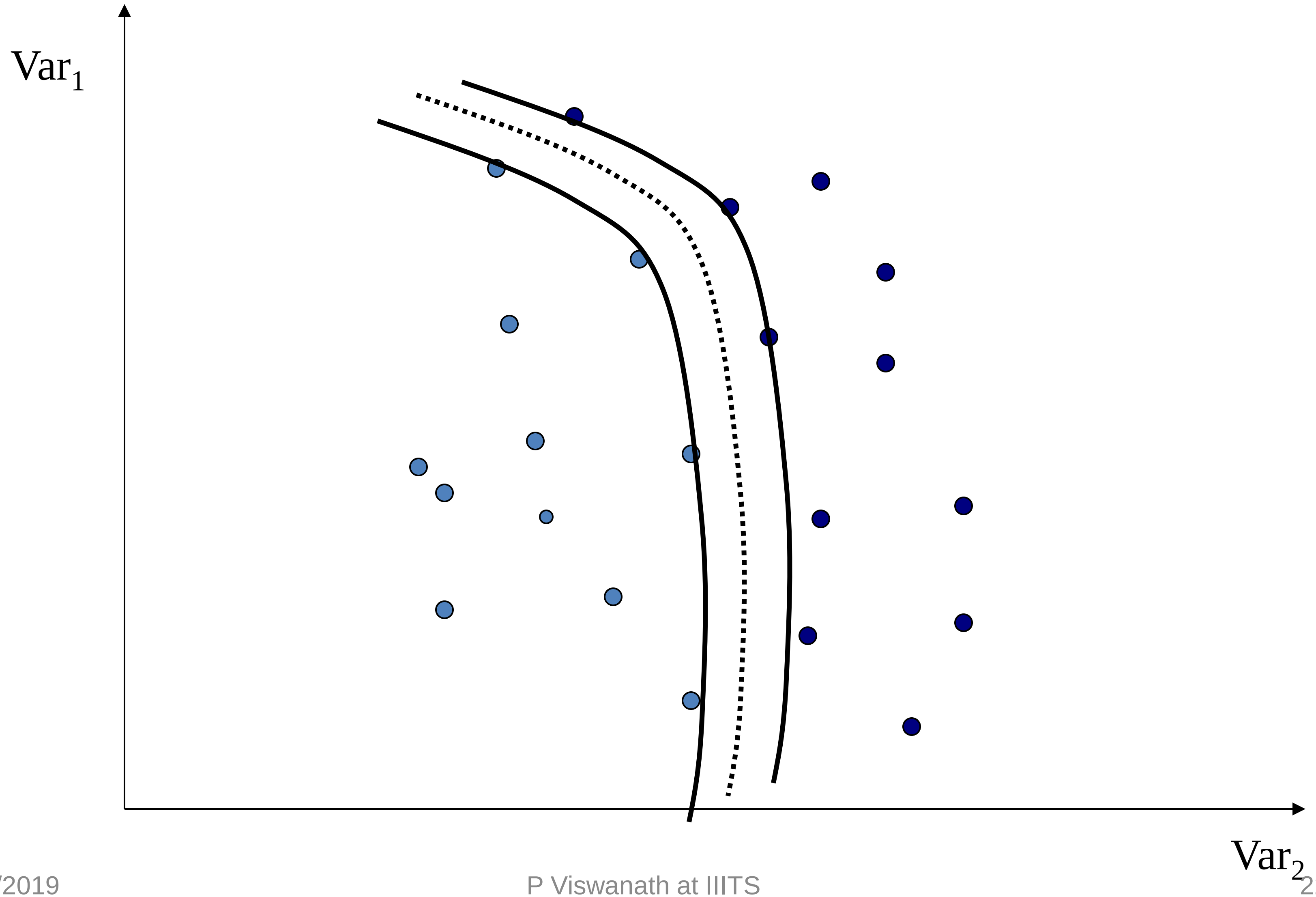
# Support Vector Machines

- 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

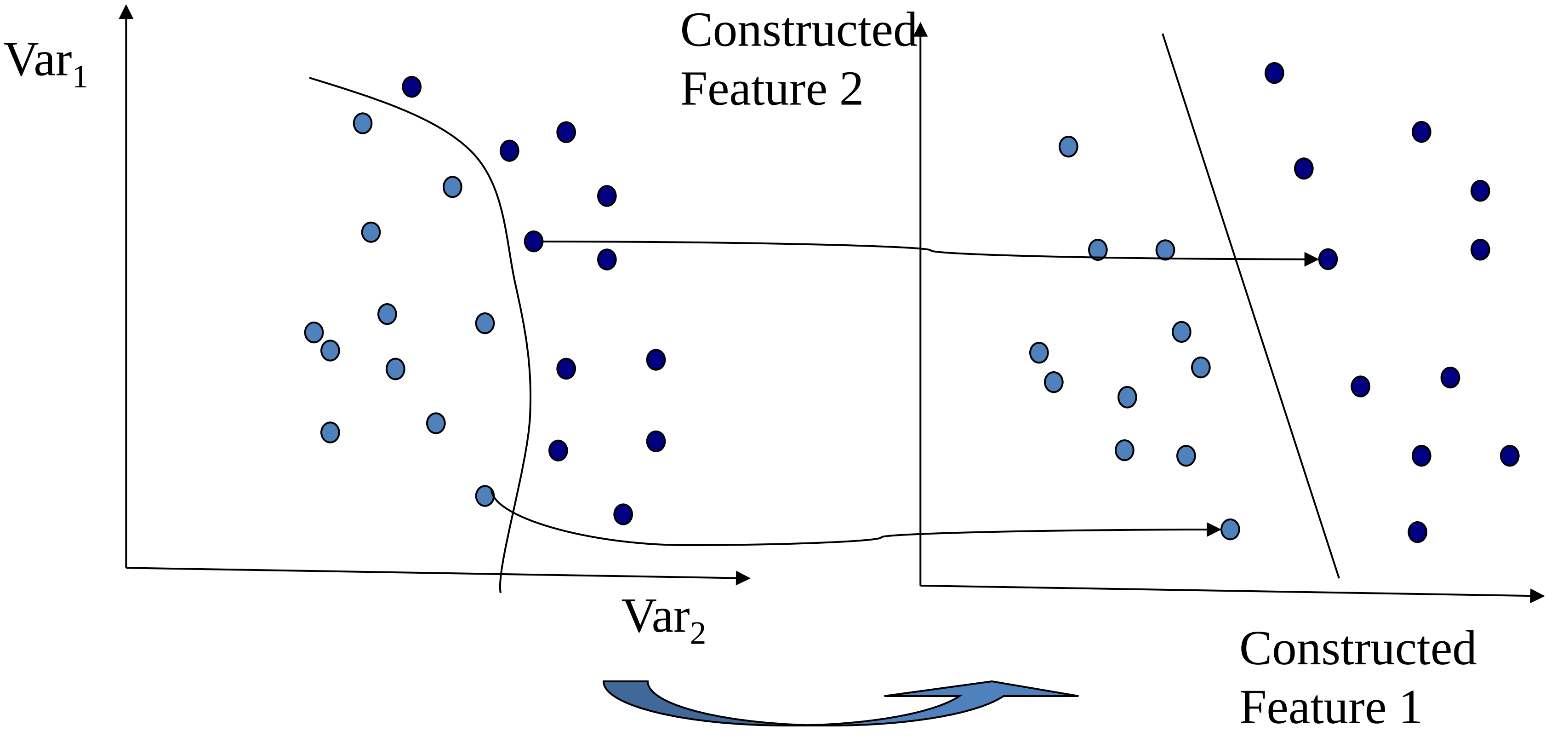
# Disadvantages of Linear Decision Surfaces



# Advantages of Non-Linear Surfaces



# Linear Classifiers in High-Dimensional Spaces




Find function $\Phi$(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:
- $\min \frac{1}{2} ||W||^2 + C \sum_i \xi_i$ $\quad$ $s.t.\ \ y_i(W \cdot \Phi(X) + b) \geq 1 - \xi_i, \forall X_i$ 
  $\xi_i \geq 0$
- 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 $\xi$ 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 $\alpha$ 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 $\Phi(X_i) \cdot \Phi(X_j)$

# 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: $K(X_i , X_j) = \Phi(X_l) \cdot \Phi(X_j)$, i.e., the image of the inner product of the data is the inner product of the images of the data
- 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

$$\operatorname{sgn}(W \cdot X + b) = \operatorname{sgn}\Big(\sum_i \alpha_i y_i K(X_i, X) + b\Big)$$

$$\text{where } b \text{ solves } \alpha_j \Big(y_j \sum_i \alpha_i y_i K(X_i, X_j) + b - 1\Big) = 0,$$

$$\text{for any } j \text{ with } \alpha_j \neq 0$$

# Examples of Kernels

- Assume we measure two quantities
- Consider the function:

$$\Phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2, x_1, x_2, 1)$$

- We can verify that:

$$K(X_1, X_2) = (X_1 \cdot X_2 + 1)^2$$

- These type of kernels are called Polynomial kernels.

# Polynomial and Gaussian Kernels

$$K(X \cdot Z) = (X \cdot Z + 1)^p$$

- is called the polynomial kernel of degree *p*.
- Another commonly used Kernel is the Gaussian (maps to an infinite dimensional space):

$$K(X \cdot Z) = \exp(-\|X - Z\| / 2\sigma^2)$$

# The Mercer Condition

- Is there a mapping $\Phi(x)$ for any given symmetric function $K(\mathrm{x},z)$? No.
- The SVM dual formulation requires calculation $K(x_i, x_j)$ for each pair of training instances. The matrix $G_{ij} = K(x_i, x_j)$ 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)

# 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
- $\nu$-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
- Kernel-PCA, Kernel-SVD

# 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 |

# 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

# Conclusions, before going in to the solution

- 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.
- B. Schölkopf. SVM and kernel methods, 2001. Tutorial given at the NIPS Conference.
- Hastie, Tibshirani, Friedman, The Elements of Statistical Learning, Springel 2001