Support Vector Machines

Some slides adapted from

- *Aliferis & Tsamardinos, Vanderbilt University*
  http://discover1.mc.vanderbilt.edu/discover/public/ml_tutorial_old/index.html

- *Rong Jin, Language Technology Institute*
  www.contrib.andrew.cmu.edu/~jin/ir_proj/svm.ppt

Support Vector Machines

- Decision surface: a hyperplane in **feature** space
- One of the most important tools in the machine learning toolbox
- 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 (weighted average of the) misclassifications

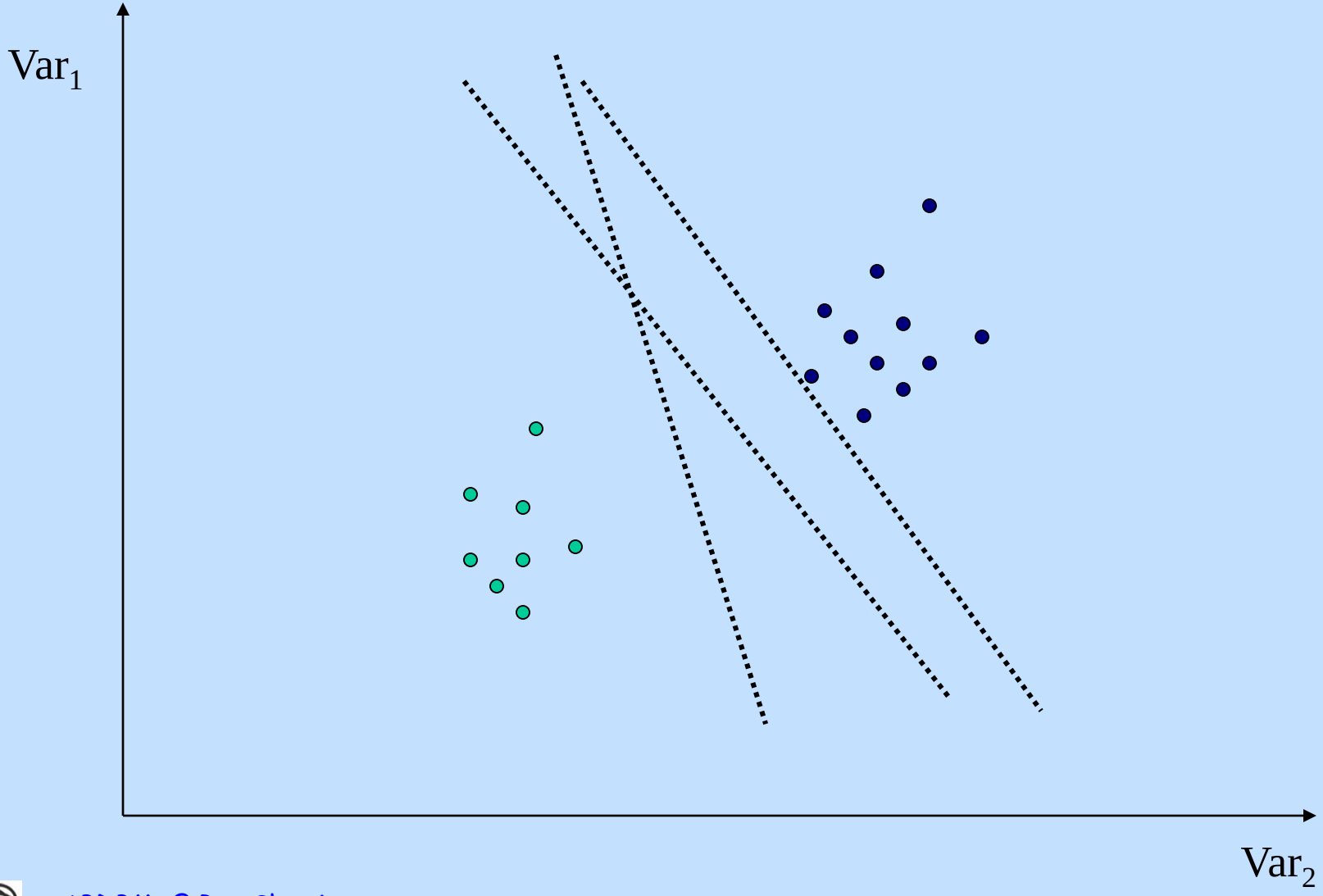
Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (taking into account that it needs to be computed efficiently): maximize margin
  2. Generalize to 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 are mapped implicitly to this space

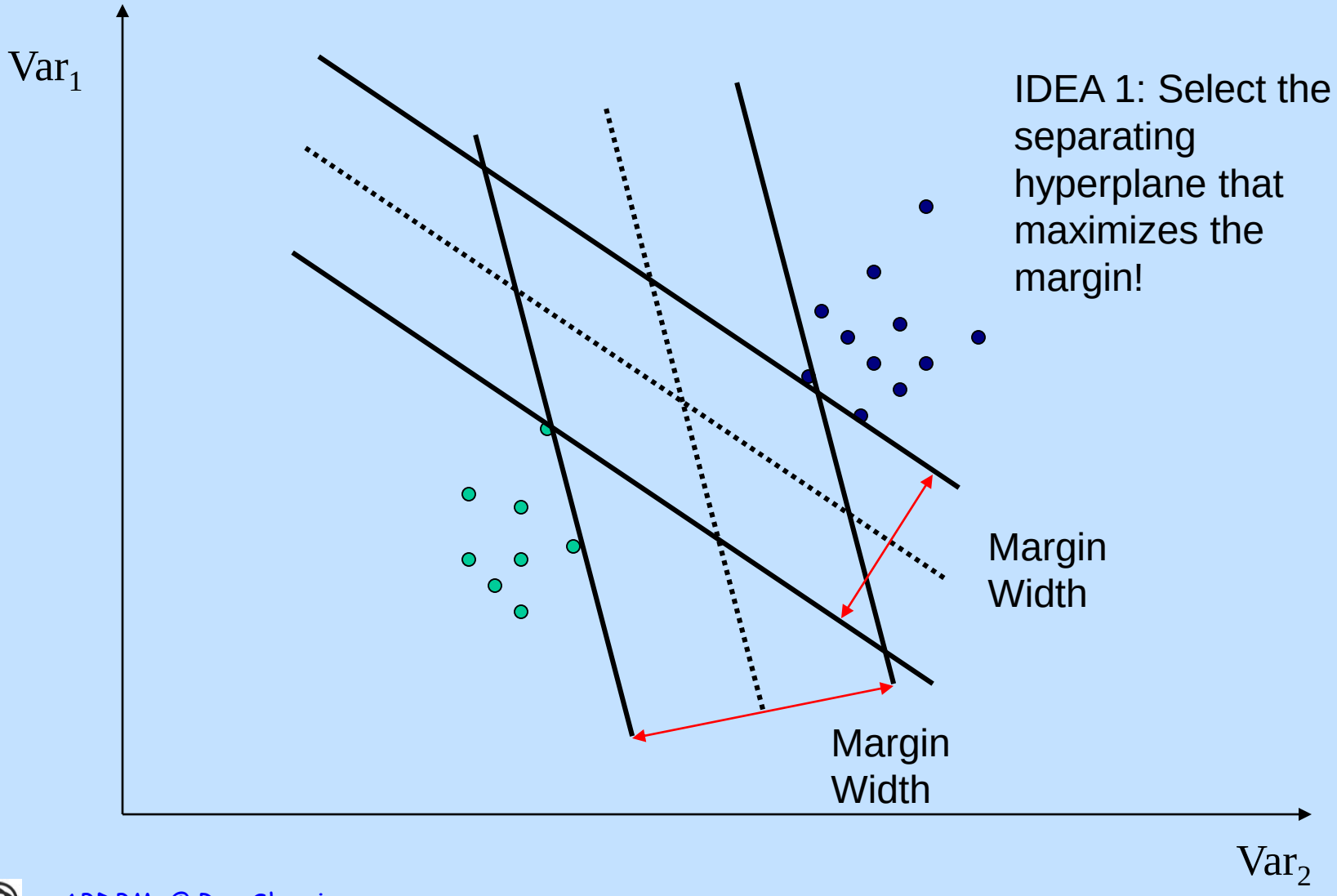
Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (taking into account that it needs to be computed efficiently): maximize margin
  2. Generalize to 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 are mapped implicitly to this space

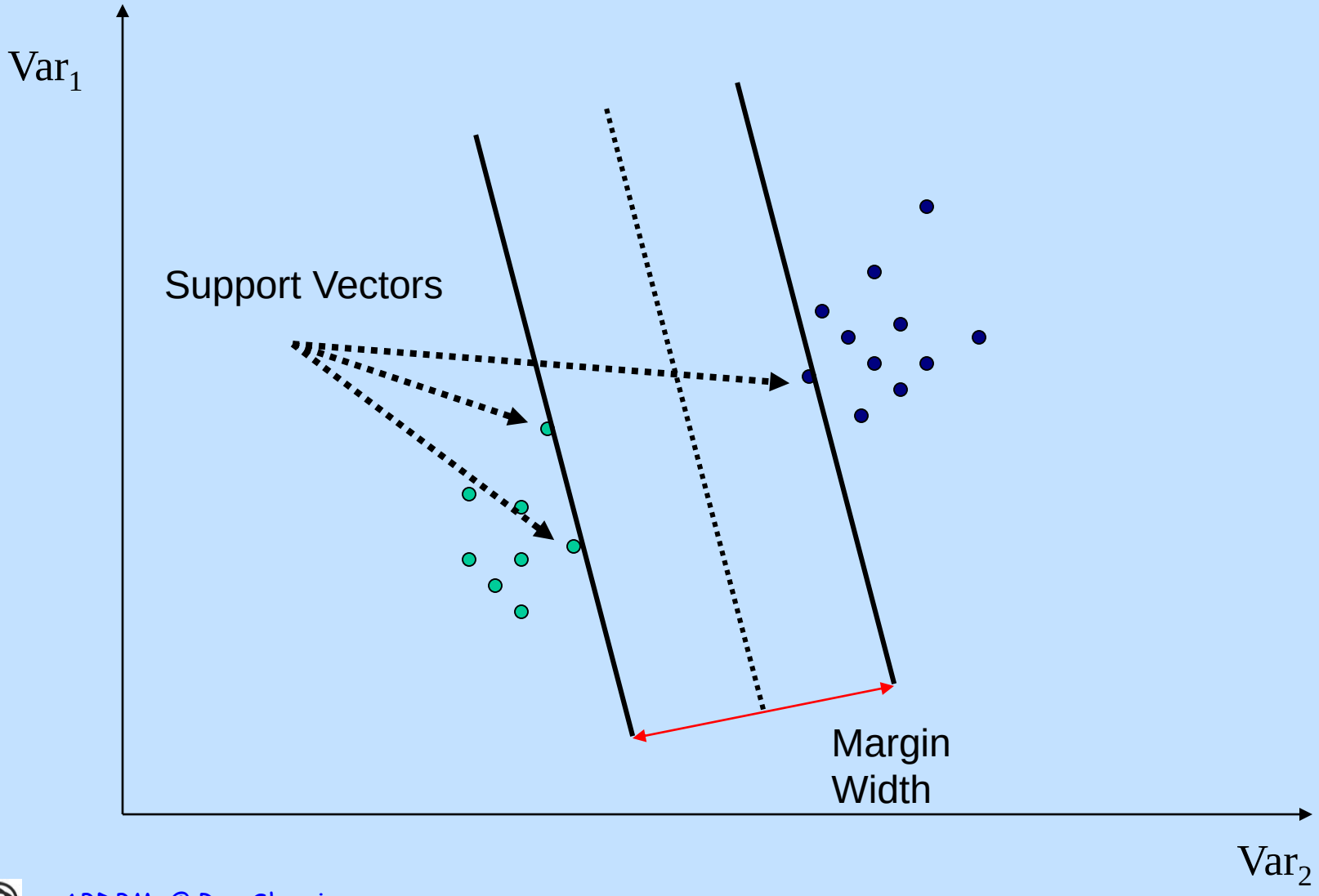
Which Separating Hyperplane to Use?



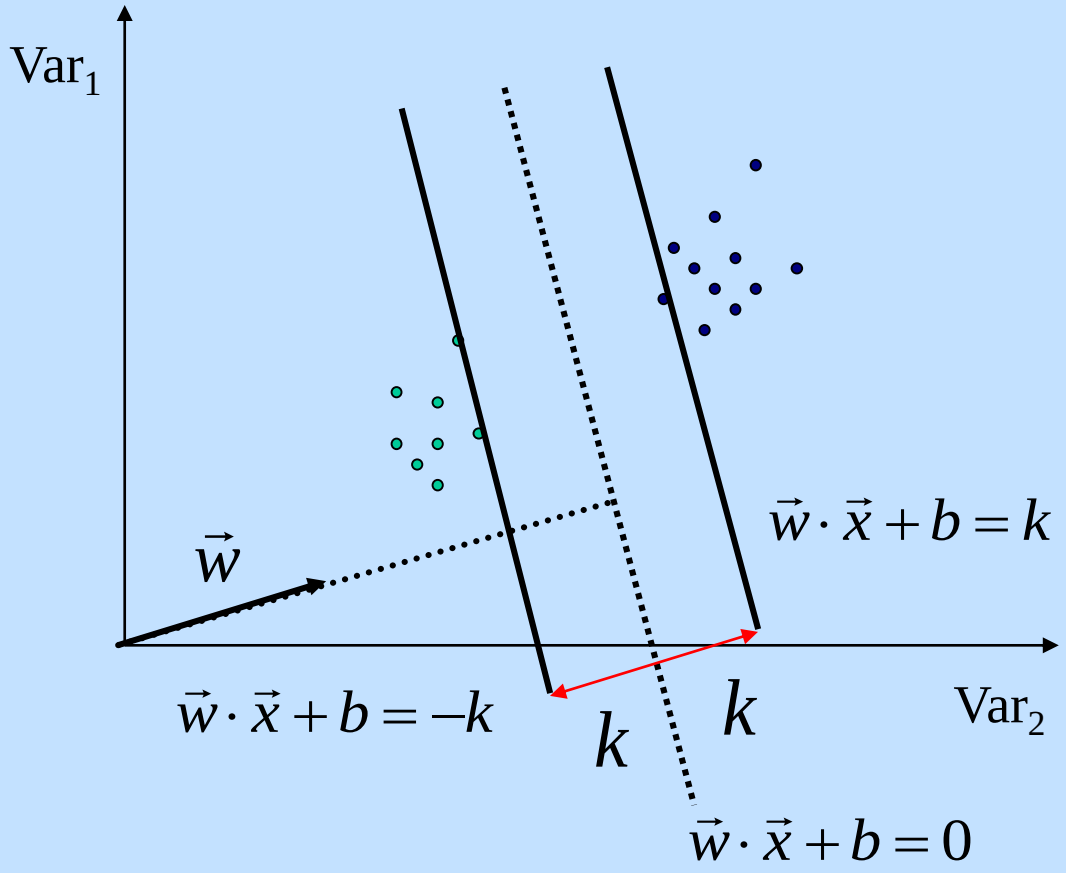
Maximizing the Margin



Support Vectors



Setting Up the Optimization Problem



The width of the margin is:

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

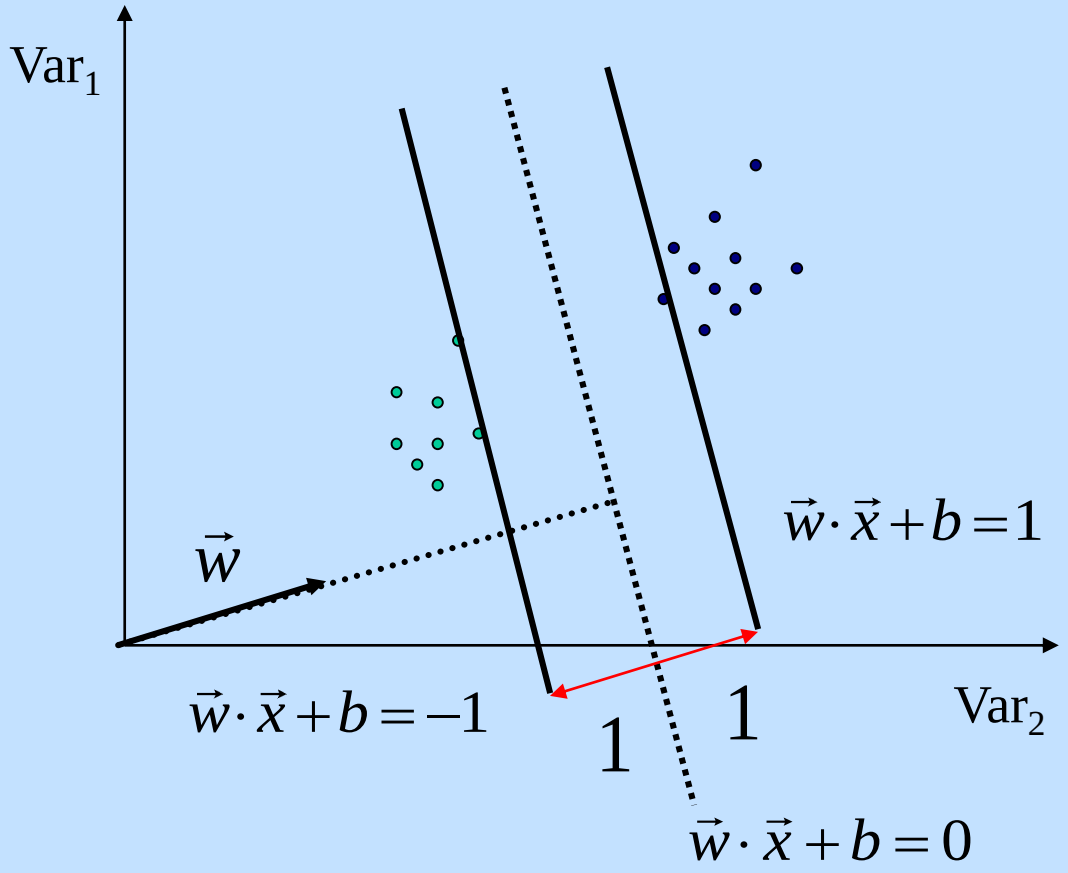
So, the problem is:

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

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

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

Setting Up the Optimization Problem



Scaling w, b so that *k=1*, the 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:

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

or

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

Linear, Hard-Margin SVM Formulation

- Find *w,b* that solve

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

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

- Quadratic program: quadratic objective, linear (in)equality constraints
- Problem is convex → there is a unique global minimum value (when feasible)
- There is also a unique minimizer, i.e. w and *b* values that provide the minimum
- No solution if the data are not linearly separable
- Objective is PD → polynomial-time soln
- Very efficient soln with modern optimization software (handles 1000s of constraints and training instances).

Lagrange multipliers

$$\underset{w,b,\alpha}{\text{Minimize}} \quad \Phi(\mathbf{w}) = \frac{1}{2} \| \mathrm{w} \|^2 - \sum \alpha_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) + \sum \alpha_i$$

$$s.t. \; \alpha_1 \geq 0, \ldots, \alpha_l \geq 0$$

- Convex quadratic programming problem
- Duality theory applies!

Dual Space

- Dual Problem

$$\text{Maximize} \quad F(\Lambda) = \Lambda \cdot 1 - \frac{1}{2} \Lambda \cdot D\Lambda$$

$$\text{subject to} \quad \Lambda \cdot \mathbf{y} = 0, \quad \Lambda \geq 0$$

$$\text{where} \quad \Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_l), \mathbf{y} = (y_1, y_2, \ldots, y_n), D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

- Representation for w

$$\mathbf{w} = \sum_{i=1}^{l} \lambda_i y_i \mathbf{x}_i$$

- Decision function

$$f(\mathbf{x}) = sign(\sum_{i=1}^{l} y_i \lambda_i (\mathbf{x} \cdot \mathbf{x}_i) + b)$$

Comments

- Representation of vector w
  - Linear combination of examples x_i
    - # parameters = # examples
  - λ_i: the importance of each examples
    - Only the points closest to the bound have $\lambda_i \neq 0$

$$\mathbf{w} = \sum_{i=1}^{l} \lambda_i y_i \mathbf{x}_i$$

- Core of the algorithm: $x \bullet x'$
  - Both matrix D and decision function require the knowledge of $x \bullet x'$

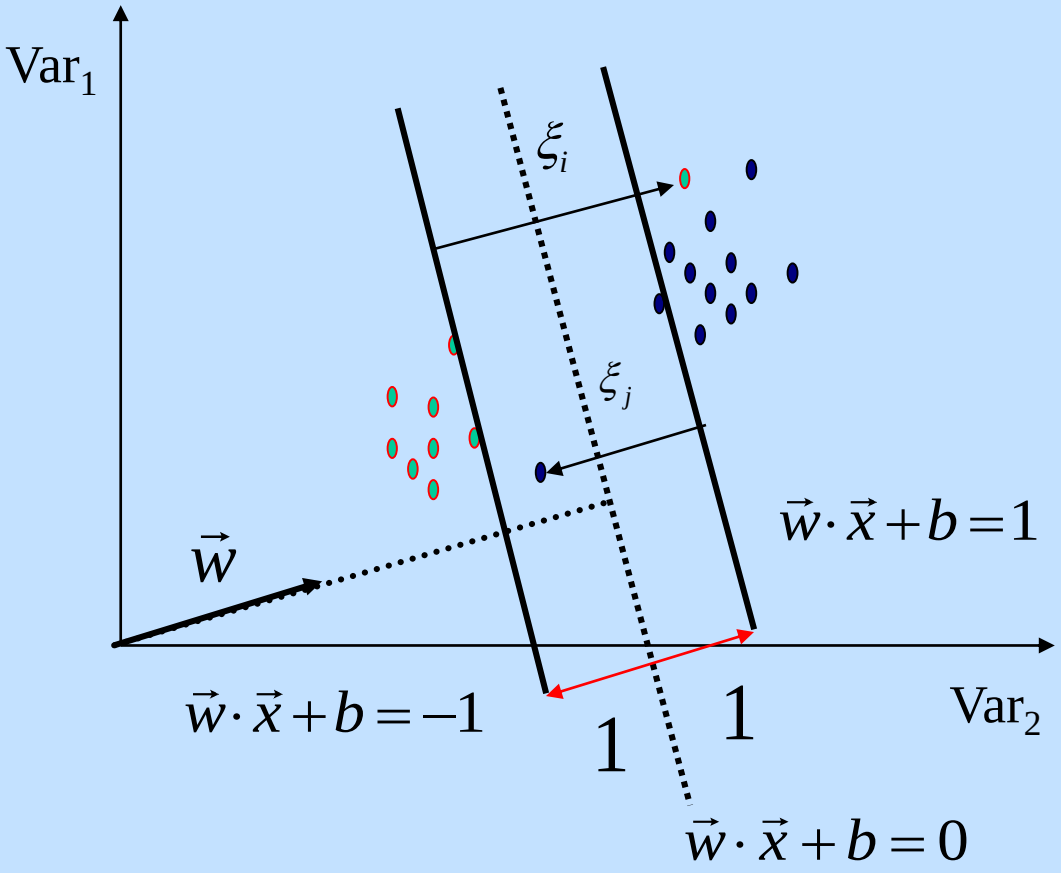
(More on this soon)

$$D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (taking into account that it needs to be computed efficiently): maximize margin
  2. Generalize to 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 are mapped implicitly to this space

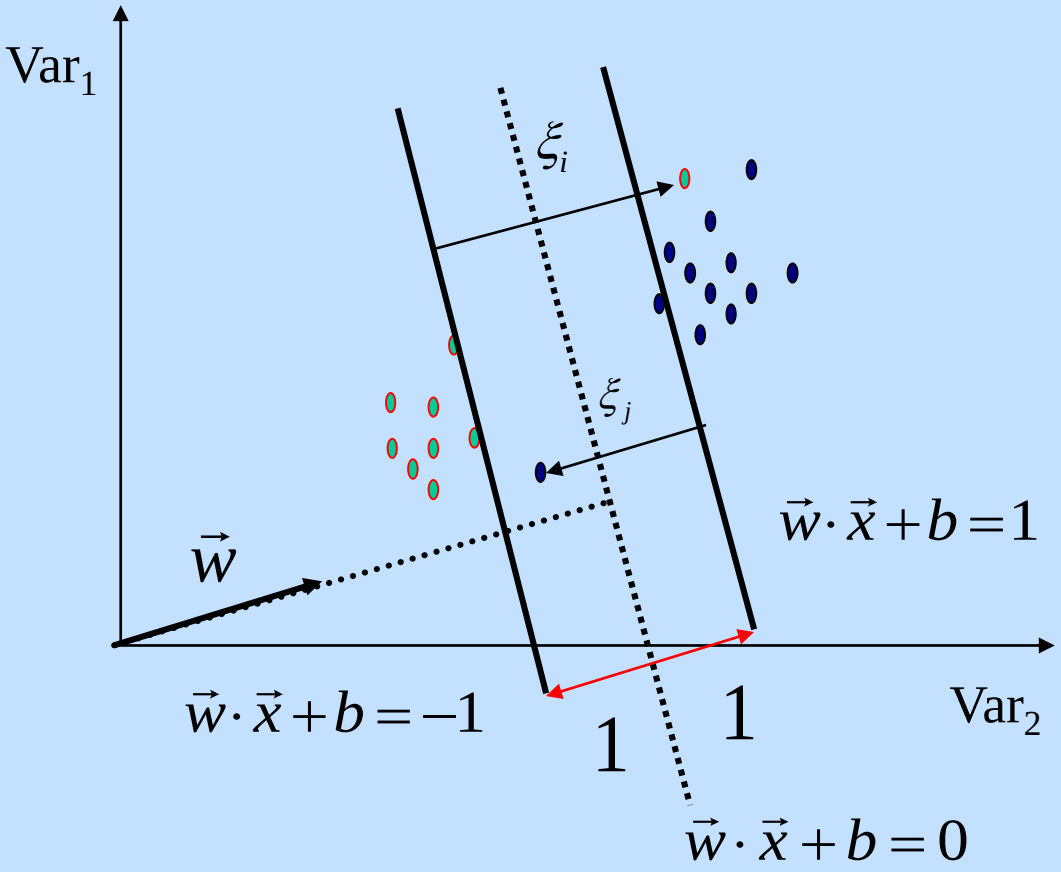
Non-Linearly Separable Data



Introduce slack variables ξ_i

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

Formulating the Optimization Problem



Constraint 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 & misclassifications

Linear, Soft-Margin SVMs

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

- Algorithm tries to keep ξ_i at zero while maximizing margin
- Alg does not minimize the *no.* of misclassifications (NP-complete problem) but the sum of distances from the margin hyperplanes
- Other formulations use ξ_i^2 instead
- C: penalty for misclassification
- As $C \to \infty$, we get closer to the hard-margin solution

Dual Space

- Dual Problem

$$\text{Maximize} \quad F(\Lambda) = \Lambda \cdot 1 - \frac{1}{2} \Lambda \cdot D\Lambda$$

$$\text{subject to} \quad \Lambda \cdot \mathbf{y} = 0$$

$$\Lambda \geq 0$$

$$\Lambda \leq C\mathbf{1}$$

$$\text{where} \quad \Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_l), \mathbf{y} = (y_1, y_2, \ldots, y_n), D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

- Only difference: upper bound C on λ_i
- Representation for w

$$\mathbf{w} = \sum_{i=1}^{l} \lambda_i y_i \mathbf{x}_i$$

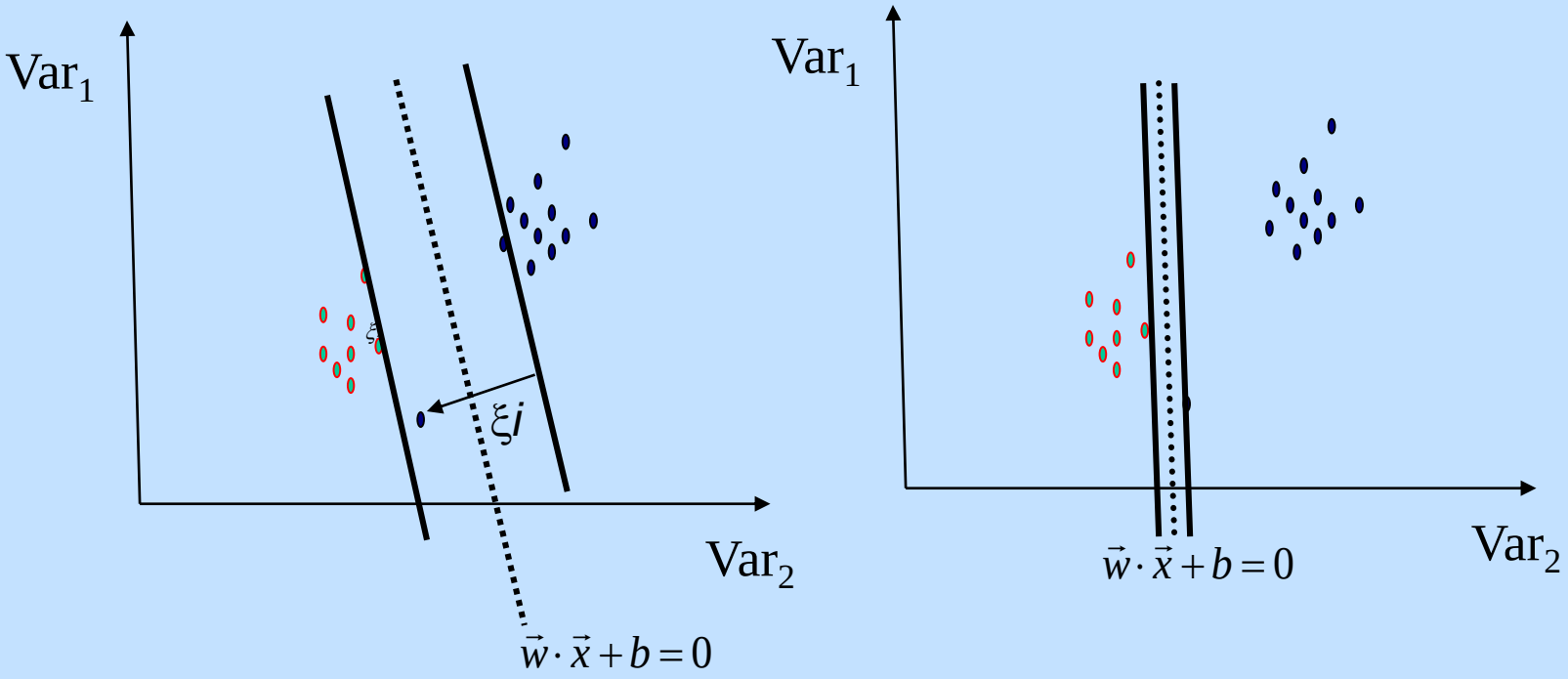
- Decision function

$$f(\mathbf{x}) = sign(\sum_{i=1}^{l} y_i \lambda_i (\mathbf{x} \cdot \mathbf{x}_i) + b)$$

Comments

- Param C
  - Controls the range of λ_i → avoids over emphasizing some examples
  - $\xi_i (C - \lambda_i) = 0$ ("complementary slackness")
  - C can be extended to be case-dependent
- Weight λ_i
  - $\lambda_i < C \rightarrow \xi_i = 0 \rightarrow$ i-th example is correctly classified → not quite important
  - $\lambda_i = C \rightarrow \xi_i$ can be nonzero → i-th training example may be misclassified → very important

Robustness of Soft vs Hard Margin SVMs



Soft Margin SVM

Hard Margin SVM

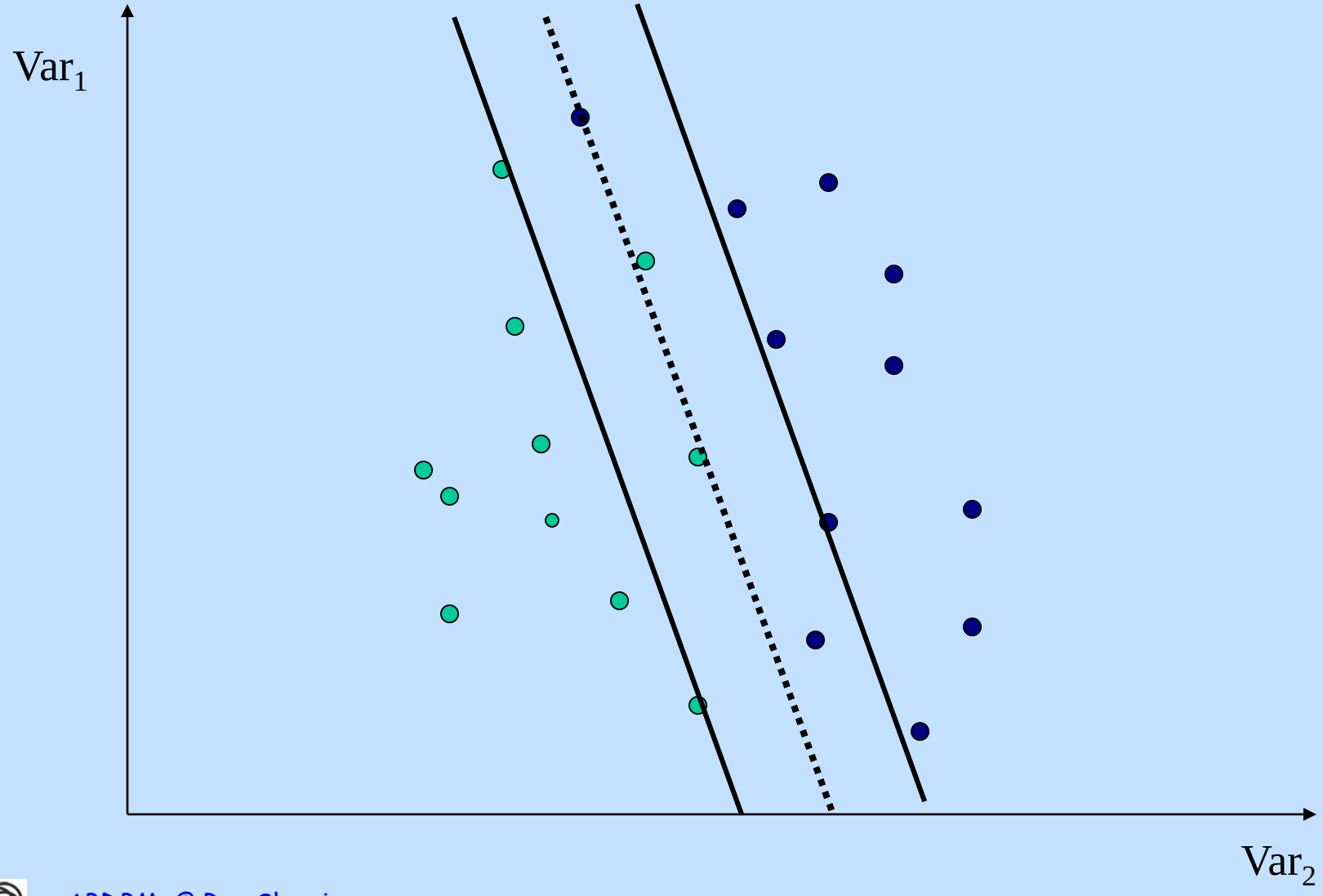
Soft vs Hard Margin SVM

- Soft-Margin always has 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)

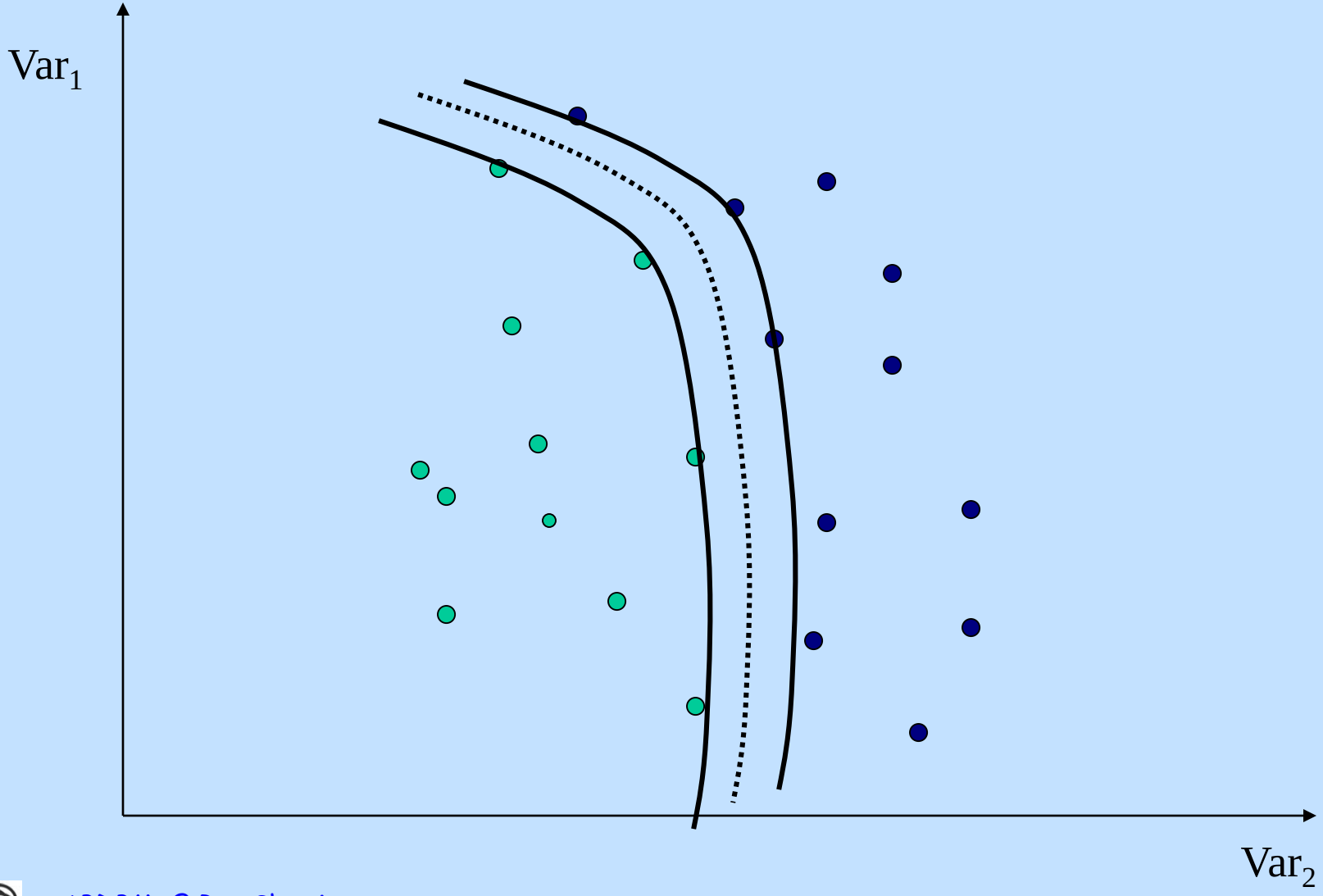
Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (taking into account that it needs to be computed efficiently): maximize margin
  2. Generalize to 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 are mapped implicitly to this space

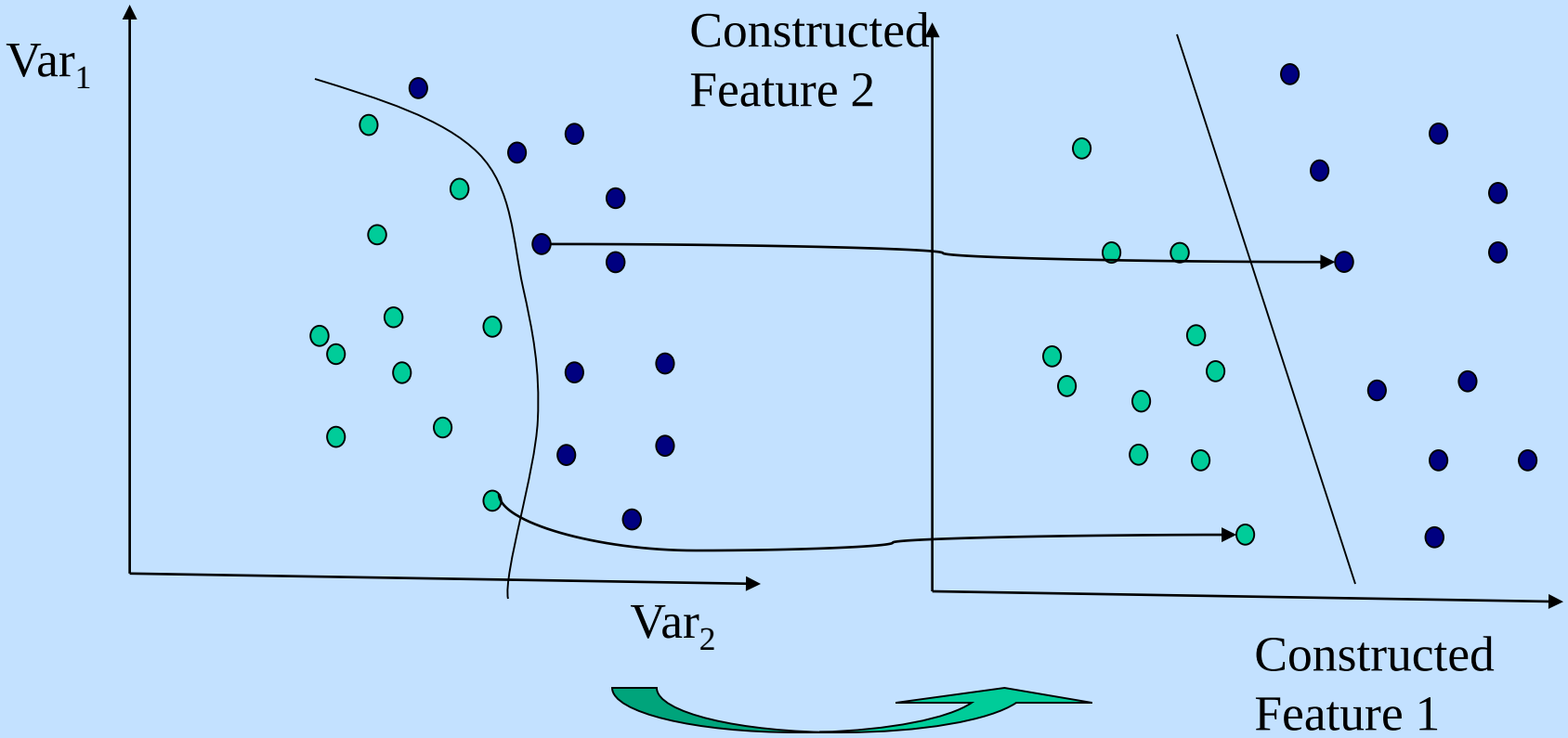
Disadvantages of Linear Decision Surfaces



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:

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

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

The Dual of the SVM Formulation

- Original SVM formulation
  - *n* inequality constraints
  - *n positivity constraints*
  - *n* number of ξ variables
- The (Wolfe) dual of this problem
  - one equality constraint
  - *n* positivity constraints
  - *n* number of α variables (Lagrange multipliers)
  - Objective function more complicated
- But: Data only appear as $\Phi(x_i) \cdot \Phi(x_j)$

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

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

$$\xi_i \geq 0$$

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

$$s.t.\ \ C \geq \alpha_i \geq 0, \forall x_i$$

$$\sum_i \alpha_i y_i = 0$$

The Kernel Trick

- $\Phi(x_i)^\dagger \cdot \Phi(x_j)$ means: map data into new space, then take the inner product of the new vectors
- Suppose we can find a function such that: $K(x_i, x_j) = \Phi(x_i)^\dagger \cdot \Phi(x_j)$, i.e., K is the inner product of the images of the data
- ➔ For training, no need to explicitly map the data into the high-dimensional space to solve the optimization problem
- How do we classify without explicitly mapping the new instances? Turns out

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

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

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

Examples of Kernels

- Assume we measure x_1,x_2 and we use the mapping:

$$\Phi :< x_1, x_2 > \rightarrow \{x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1\}$$

- Consider the function:

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

- Then:
- $\phi(x)^t\phi(z) = x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1 = (x_1z_1 + x_2z_2 + 1)^2 = (x \cdot z + 1)^2 = K(x, z)$

Polynomial and Gaussian Kernels

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

is called the polynomial kernel of degree *p*.

- For *p=2*, with 7,000 genes using the kernel once: inner product with 7,000 terms, squaring
- Mapping explicitly to the high-dimensional space: calculating ~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 common option: Gaussian kernel (maps to *l* dimensional space with *l*=no of training points):

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

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 matrix $G_{ij} = K(x_i,x_j)$ is called the Gram matrix
- Theorem (Mercer 1908): There is a feature space $\Phi(x)$ iff the Kernel is such that G is positive-semi definite
- Recall: M PSD iff $\forall z \neq 0$ $z^T M z > 0$ iff M has non-negative eigenvalues

Support Vector Machines

- Three main ideas:
  1. Define what an optimal hyperplane is (taking into account that it needs to be computed efficiently): maximize margin
  2. Generalize to 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 are mapped implicitly to this space

Complexity
(for one implementation, Burges 98)

Notation: l training pts of dimension d, N support vectors ($N \leq l$)

- When most SVs are not at the upper bound:
  - $O(N^3+N^2l+Ndl)$ if $N<<l$
  - $O(N^3+nl+Ndl)$ if $N \sim l$
- When most SVs are at the upper bound:
  - $O(N^2 + Ndl)$ if $N<<l$
  - $O(dl^2)$ if $N \sim l$

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 allow different cost of misclassification for different classes
- Kernels suitable for sequences of strings, or other specialized kernels

Feature Selection with SVMs

- Recursive Feature Elimination
  - Train a linear SVM
  - Remove the x% of variables with the lowest weights (those variables affect classification the least)
  - Retrain the SVM with remaining variables and repeat until classification quality is reduced
- Very successful
- Other formulations exist where minimizing the number of variables is folded into the optimization problem
- Similar algs for non-linear SVMs
- Quite successful

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

Conclusions

- SVMs formulate learning as a mathematical program taking advantage of the rich theory in optimization
- SVM uses kernels to map indirectly to extremely high dimensional spaces
- SVMs are extremely successful, robust, efficient, and versatile, and have a good theoretical basis

Vladimir Vapnik

- Vladimir Naumovich Vapnik is one of the main developers of Vapnik–Chervonenkis theory. He was born in the Soviet Union. He received his master's degree in mathematics at the Uzbek State University, Samarkand, Uzbek SSR in 1958 and Ph.D in statistics at the Institute of Control Sciences, Moscow in 1964. He worked at this institute from 1961 to 1990 and became Head of the Computer Science Research Department.

  At the end of 1990, he moved to the USA and joined the Adaptive Systems Research Department at AT&T Bell Labs in Holmdel, New Jersey. The group later became the Image Processing Research Department of AT&T Laboratories when AT&T spun off Lucent Technologies in 1996. Vapnik Left AT&T in 2002 and joined NEC Laboratories in Princeton, New Jersey, where he currently works in the Machine Learning group. He also holds a Professor of Computer Science and Statistics position at Royal Holloway, University of London since 1995, as well as an Adjunct Professor position at Columbia University, New York City since 2003. He was inducted into the U.S. National Academy of Engineering in 2006. He received the 2008 Paris Kanellakis Award.
- While at AT&T, Vapnik and his colleagues developed the theory of the support vector machine. They demonstrated its performance on a number of problems of interest to the machine learning community, including handwriting recognition.

http://en.wikipedia.org/wiki/Vladimir_Vapnik

Suggested Further Reading

- http://www.kernel-machines.org/tutorial.html
- http://www.svms.org/tutorials/ - many tutorials
- **C. J. C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition." *Knowledge Discovery and Data Mining*, 2(2), 1998.**
- E. Osuna, R. Freund, and F. Girosi. "Support vector machines: Training and applications." Technical Report AIM-1602, MIT A.I. Lab., 1996.
- 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

Analysis of microarray GE data using SVM

Brown, Grundy, Lin, Cristianini, Sugnet, Furey, Ares Jr., Haussler

PNAS 97(1) 262-7 (2000)

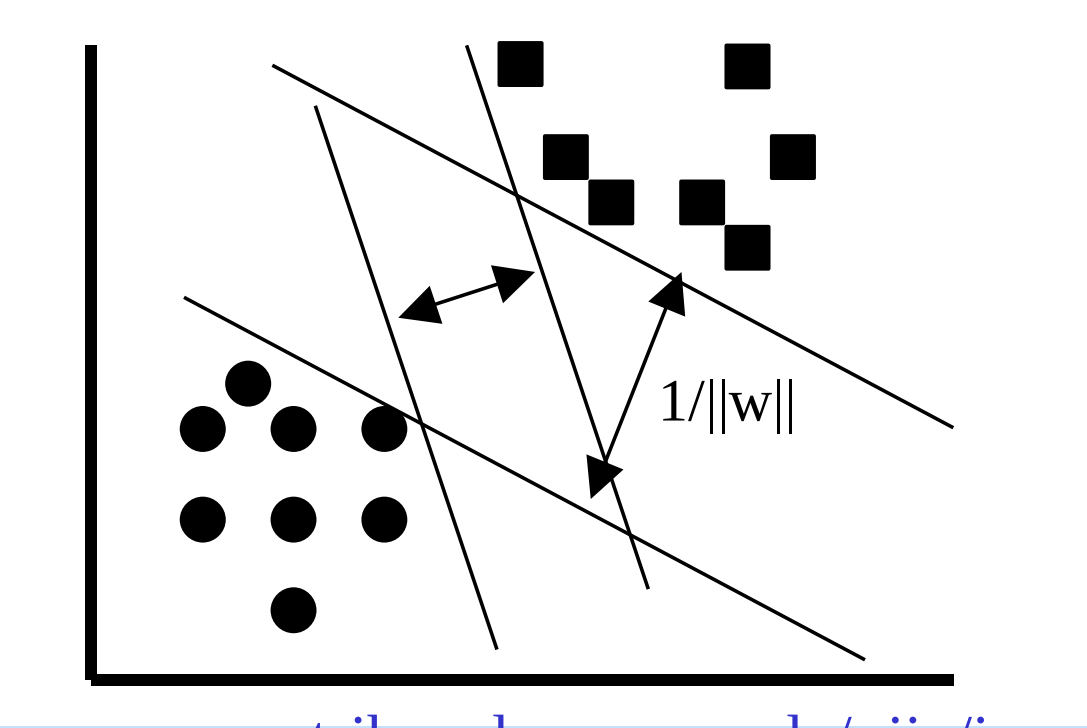
Data

- Expression patterns of n=2467 annotated yeast genes over m=79 different conditions
- Six gene functional classes: 5 related to transcript levels, tricarboxylic acid (TCA) cycle, respiration, cytoplasmic ribosomes, proteasome, histones, and 1 unrelated (control) helix-turn-helix proteins.
- For gene x, condition i:
  - E_i level of x in tested condition
  - R_i level of x in reference condition
- Normalized pattern $(X_1,\ldots,X_m)$ of gene x:

$$X_i = \log(E_i/R_i)/(\Sigma_k \log^2(E_k/R_k))^{0.5}$$

Goal

- Classify genes based on gene expression
- Tried SVM and other classifiers



www.contrib.andrew.cmu.edu/~jin/ir_proj/

Kernel functions used

- Simplest : $K(X,Y)=X\cdot Y+1$ (dot product; linear kernel)
- Kernel of degree d: $K(X,Y)=(X\cdot Y+1)^d$
- Radial basis (Gaussian) kernel:

  $\exp(-||X-Y||^2/2\alpha^2)$
- n^+ / n^- : no. of positive / negative examples
- Problem: $n^+ << n^-$
- Overcoming imbalance: modify K's diagonal:
  $K_{ij}=K(X^i,X^j)+c/n^+$ for positive ex,
  $K_{ij}=K(X^i,X^j)+c/n^-$ for negative ex

Measuring performance

True / Classifier	+	-
+	TP	FP
-	FN	TN

- The imbalance problem: very few positives
- Performance of method M: C(M) =FP+2FN
- C(N) = cost of classifying all as negatives
- S(M) =C(N)-C(M) (how much we save by the classifier).
- 3-way cross validation: 2/3 learn, 1/3 test

Results - TCA class

Method	FP	FN	TP	TN	S(M)
D-p-1-SVM	18	5	12	2,432	6
D-p-2-SVM	7	9	8	2,443	9
D-p-3-SVM	4	9	8	2,446	12
Radial-SVM	5	9	8	2,445	11
Parzen	4	12	5	2,446	6
FLD	9	10	7	2,441	5
C4.5	7	17	0	2,443	-7
MOC1	3	16	1	2,446	-1

D-p-i-SVM: dot product kernel, degree i

Other methods used: Parzen windows, Fisher linear discriminant, C4.5+MOC1: decision trees

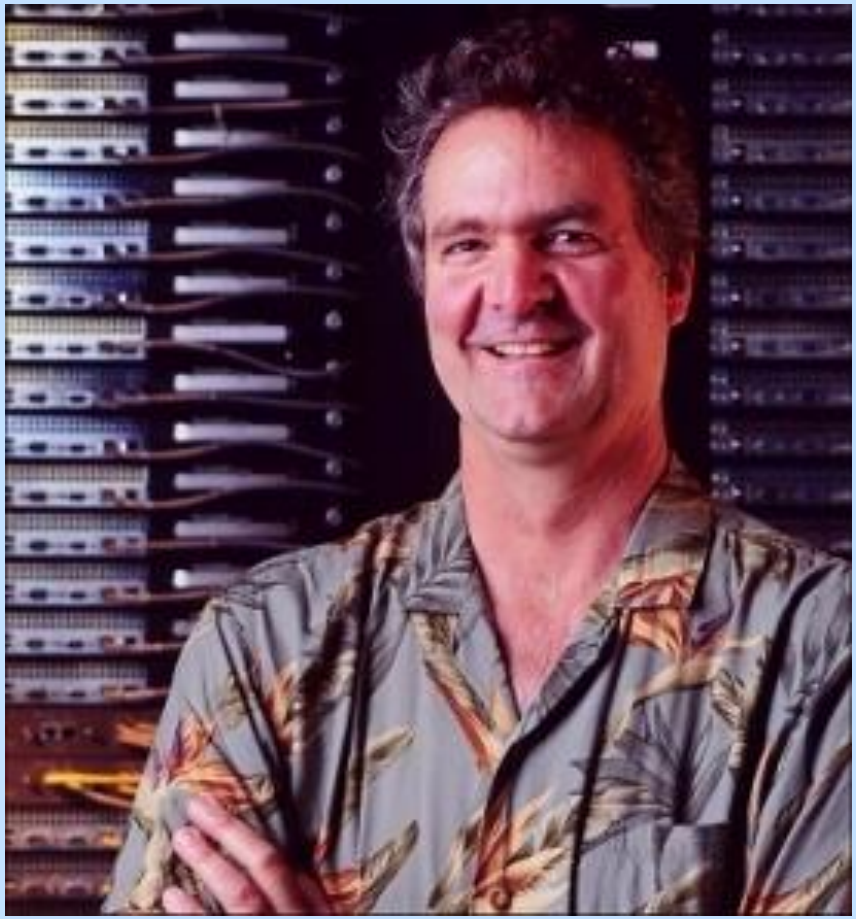
Results: Ribo Class

Method	FP	FN	TP	TN	S(M)
D-p-1-SVM	14	2	119	2,332	224
D-p-2-SVM	9	2	119	2,337	229
D-p-3-SVM	7	3	118	2,339	229
Radial-SVM	6	5	116	2,340	226
Parzen	6	8	113	2,340	220
FLD	15	5	116	2,331	217
C4.5	31	21	100	2,315	169
MOC1	26	26	95	2,320	164

Results: Summary

- SVM outperformed the other methods
- Either high-dim dot-product or Gaussian kernels worked best
- Insensitive to specific cost weighting
- Consistently misclassified genes require special attention
- Does not always reflect protein levels and post-translational modifications
- Can use classifiers for functional annotation

David Haussler



Gene Selection via the BAHSIC Family of Algorithms

Le Song, Justin Bedo,
Karsten M. Borgwardt, Arthur Gretton, Alex Smola

ISMB 07

Testing

- 15 two-class datasets (mostly cancer), 2K-25K genes, 50-300 samples
- 10-fold cross validation
- Selected the 10 top features according to each method
  - pc=Pearson's correlation, snr=signal-to-noise ratio, pam=shrunken centroid, t=t-statistics, m-t = moderated t-statistics, lods=B-statistics, lin=centroid, RBF= SVM w Gaussian kernel, rfe=SVM recursive feature elimination, l1=l_1 norm SVM, mi=mutual information)
- Selection method: RFE: Train, remove 10% of features that are least relevant, repeat.

Classification error %

Overlap btw the 10 genes selected in each fold

Linear kernel has best overall performance

	BAHSIC family									Others		
Ref.#	pc	sur	pam	t	m-t	lods	lin	RBF	dis	rfe	l1	mi
1	12.7|3	11.4|3	11.4|**4**	12.9|3	12.9|**4**	12.9|**4**	15.5|3	19.1|1	13.9|2	14.3|0	**7.7**|0	26.1|0
2	33.2|1	33.9|2	33.9|1	**29.5**|1	29.5|1	**27.8**|1	32.9|2	31.5|**3**	32.8|2	34.2|0	32.5|1	29.9|0
3	37.4|0	37.4|0	37.4|0	**34.6**|**6**	**34.6**|**6**	**34.6**|**6**	37.4|1	37.4|0	37.4|0	37.4|0	37.4|0	36.4|0
4	41.6|0	**38.8**|0	41.6|0	40.7|**1**	**40.7**|0	**37.8**|0	41.6|0	41.6|0	**39.7**|0	41.6|0	41.6|0	40.6|0
5	**27.8**|0	**26.7**|0	**27.8**|0	**26.7**|**2**	**26.7**|**2**	**26.7**|**2**	**27.8**|0	**27.8**|0	**27.6**|0	**27.8**|0	**27.8**|0	**27.8**|0
6	30.0|2	**25.0**|0	31.7|0	**25.0**|**5**	**25.0**|**5**	**25.0**|**5**	30.0|0	31.7|0	30.0|1	30.0|0	33.3|0	33.3|0
7	**2.0**|**6**	**2.0**|5	**2.0**|5	28.7|4	26.3|4	26.3|4	**2.0**|3	**2.0**|4	30.0|0	**2.0**|0	**2.0**|0	**2.0**|2
8	3.3|3	**0.0**|4	**0.0**|4	**0.0**|4	3.3|**6**	3.3|**6**	3.3|2	3.3|1	6.7|2	**0.0**|0	3.3|0	6.7|1
9	**10.0**|**6**	**10.0**|**6**	**8.7**|4	34.0|5	37.7|**6**	37.7|**6**	12.0|3	**10.0**|5	12.0|1	**10.0**|0	17.0|1	12.0|3
10	**16.0**|2	**18.0**|2	**14.0**|2	**14.0**|8	22.0|**9**	22.0|**9**	**16.0**|2	**16.0**|0	**18.0**|0	32.5|0	**14.0**|0	20.5|1
11	12.9|5	12.9|5	12.9|5	19.5|0	22.1|0	33.6|0	**11.2**|4	**9.5**|**6**	16.0|4	19.0|0	17.4|0	**11.2**|4
12	30.3|2	36.0|2	31.3|2	26.7|**3**	35.7|0	35.7|0	**18.7**|1	35.0|0	33.0|1	29.7|0	30.0|0	**23.0**|2
13	8.4|5	11.1|0	7.0|5	22.1|3	27.9|**6**	15.4|1	7.0|2	9.6|0	11.1|0	**4.3**|1	**5.5**|2	7.0|4
14	20.8|1	20.8|1	20.2|0	20.8|**3**	20.8|**3**	20.8|**3**	20.8|0	20.2|0	**19.7**|0	20.8|0	20.8|1	**19.1**|1
15	**0.0**|7	0.7|1	**0.0**|5	4.0|1	0.7|**8**	0.7|**8**	**0.0**|3	**0.0**|2	2.0|2	**0.0**|1	**0.0**|1	**0.0**|7
best	5|2	7|1	6|1	6|6	4|10	5|9	6|0	6|2	4|0	6|0	6|0	6|0
ℓ_2	6.9	20.9	17.3	43.5	50.5	50.3	**13.2**	22.9	35.4	26.3	19.7	23.5

L2 dist from best

times alg was best

Multiclass datasets

- In a similar comparison on 13 multiclass datasets, linear kernel was again best.

Rules of thumb

- Always apply the linear kernel for general purpose gene selection
- Apply a Gaussian Kernel if nonlinear effects are present, such as multimodality or complementary effects of different genes
- Not a big surprise, given the high dimension of microarray datasets, but point driven home by broad experimentation.