TEMA 7a:
Support Vector Machines

Outline

- A brief history of SVM
- Large-margin linear classifier
  - Linear separable
  - Nonlinear separable
- Creating nonlinear classifiers: kernel trick
- A simple example
- Application to Text Categorization
- Discussion on SVM
- Conclusion

History of SVM

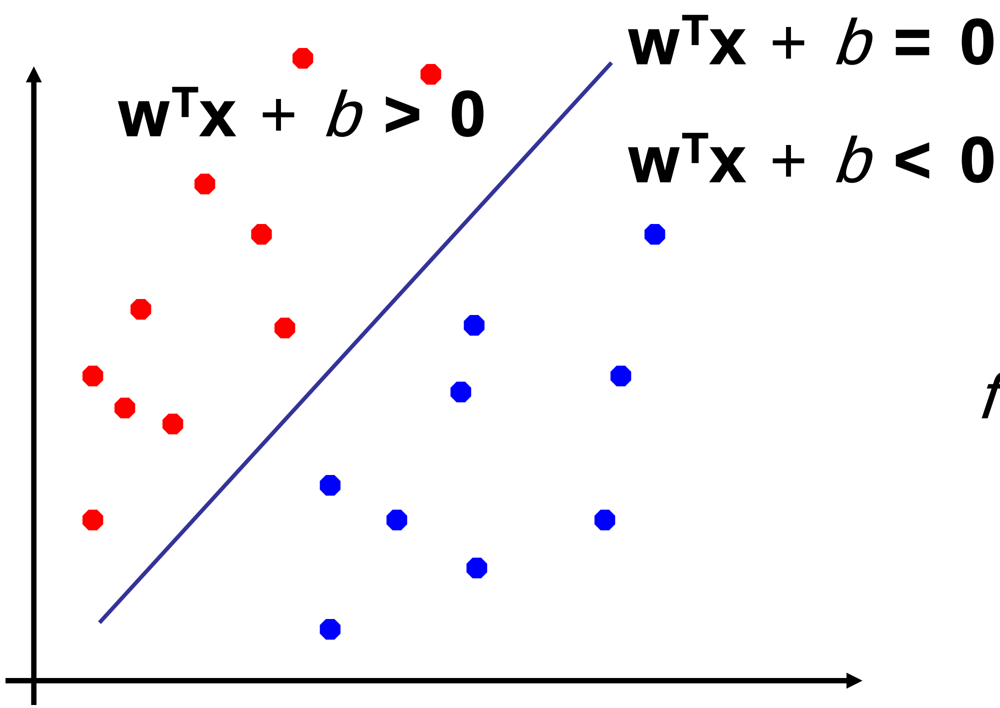
- SVM is related to statistical learning theory [3]
- SVM was first introduced in 1992 [1]
- SVM becomes popular because of its success in handwritten digit recognition
  - 1.1% test error rate for SVM. This is the same as the error rates of a carefully constructed neural network, LeNet 4.
    - See Section 5.11 in [2] or the discussion in [3] for details
- SVM is now regarded as an important example of “kernel methods”, one of the key area in machine learning

[1] B.E. Boser *et al*. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
[2] L. Bottou *et al*. Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 2, pp. 77-82.
[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

SVM: LARGE-MARGIN LINEAR CLASSIFIER

Perceptron Revisited: Linear Separators

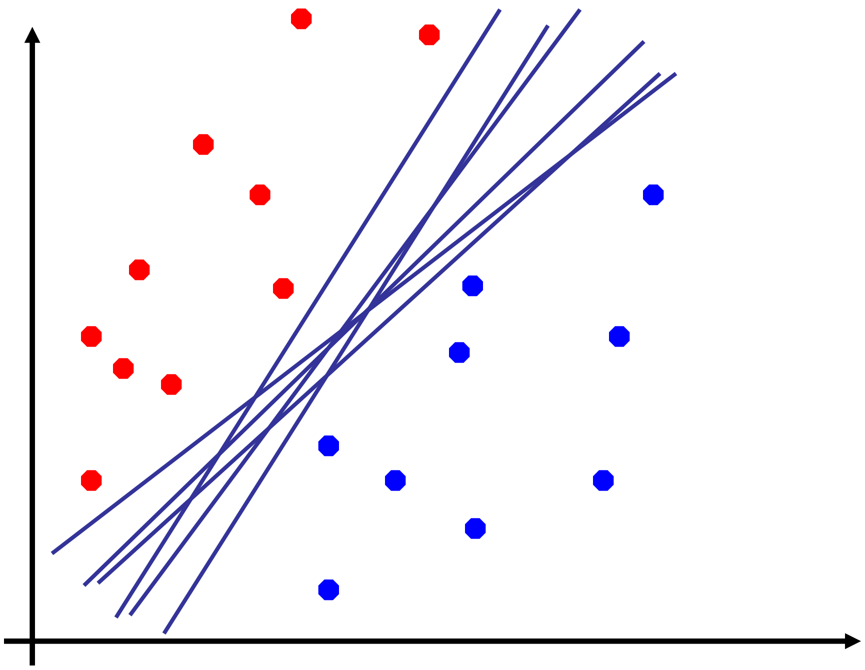
- Binary classification can be viewed as the task of separating classes in feature space:



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

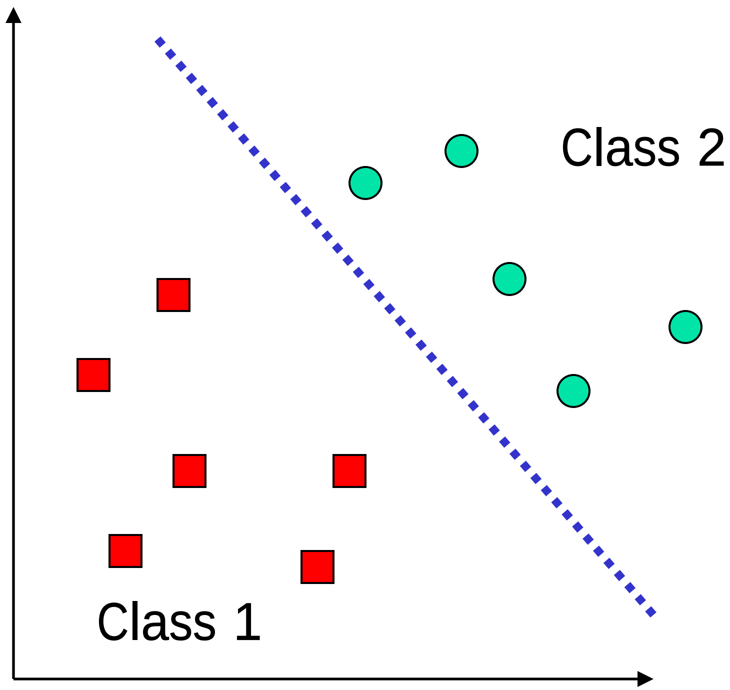
Linear Separators

- Which of the linear separators is optimal?

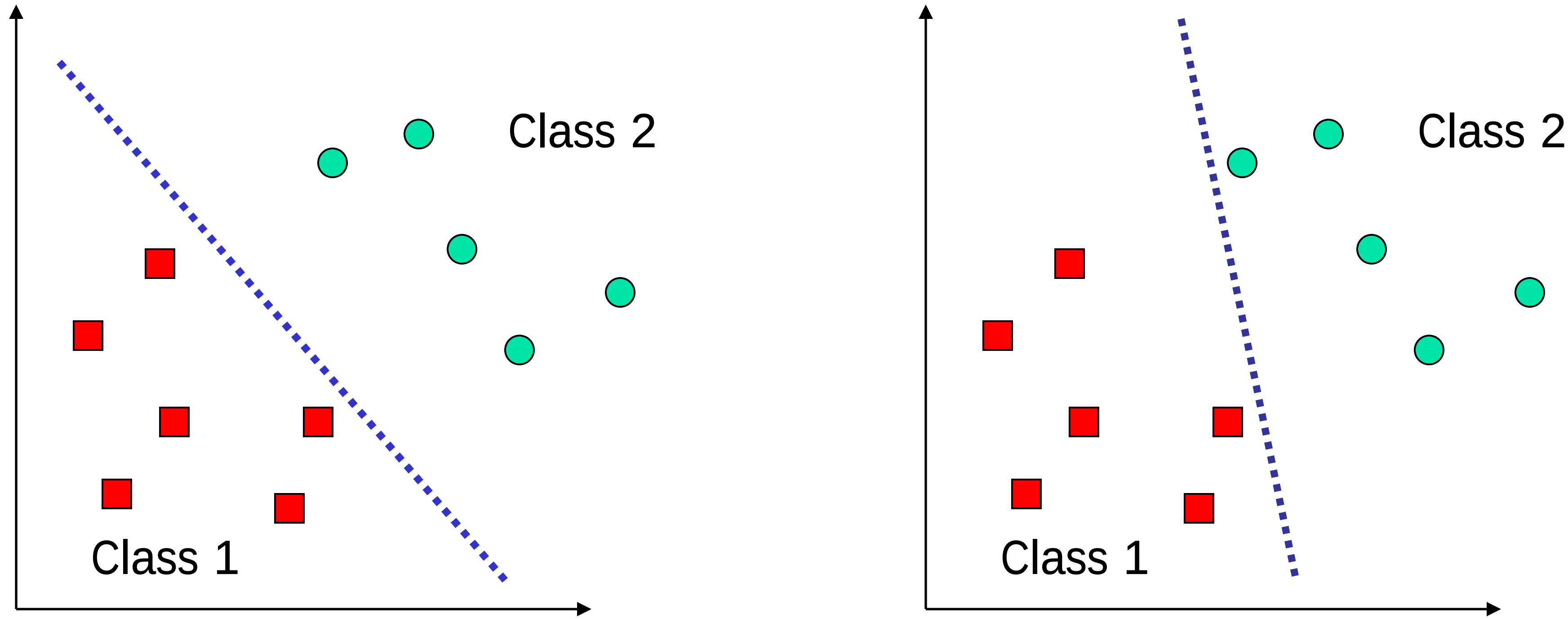


What is a good Decision Boundary?

- Consider a two-class, linearly separable classification problem
- Many decision boundaries!
  - The Perceptron algorithm can be used to find such a boundary
  - Different algorithms have been proposed (DHS ch. 5)
- Are all decision boundaries equally good?

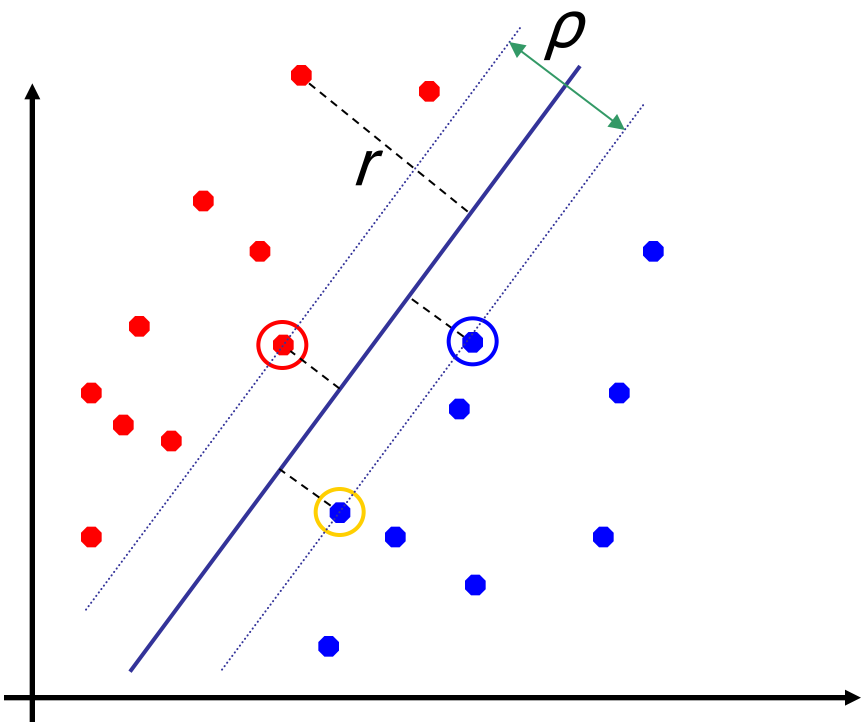


Examples of Bad Decision Boundaries



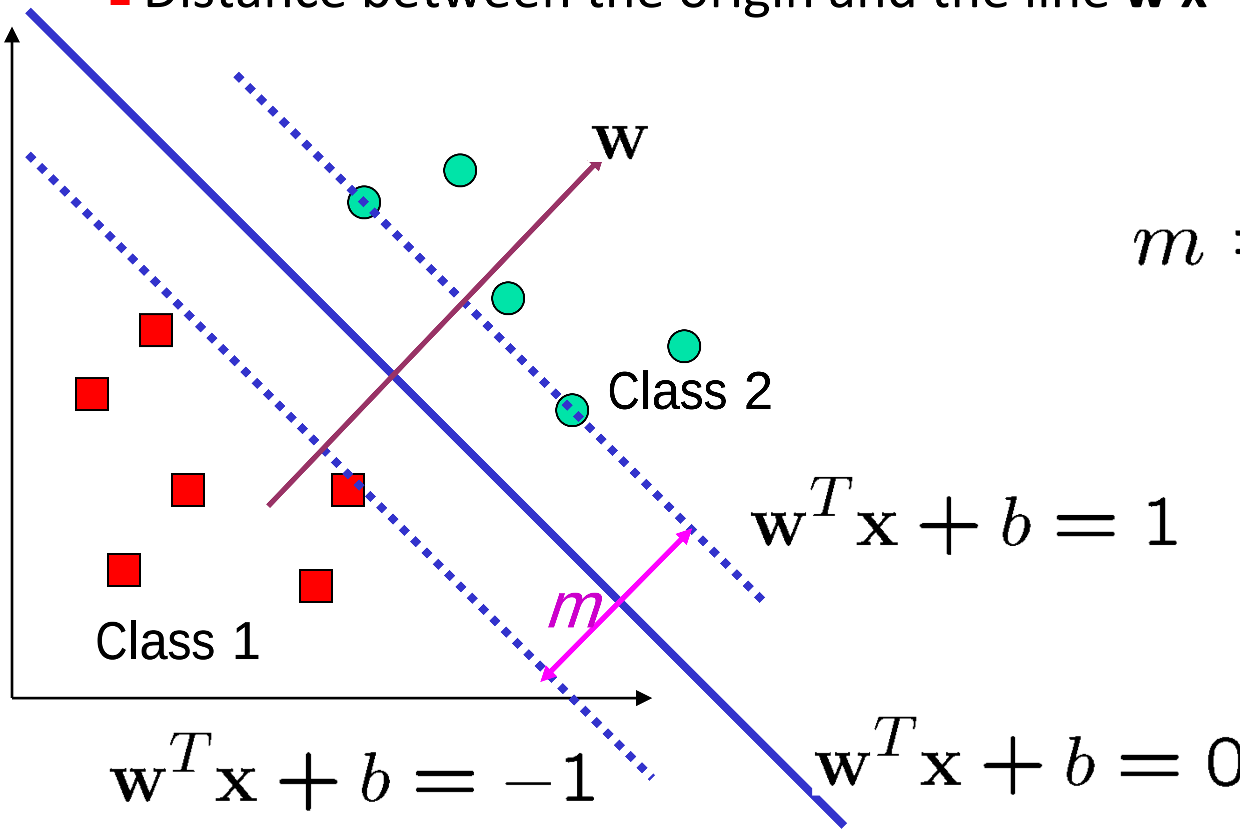
Classification Margin

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



Large-margin Decision Boundary

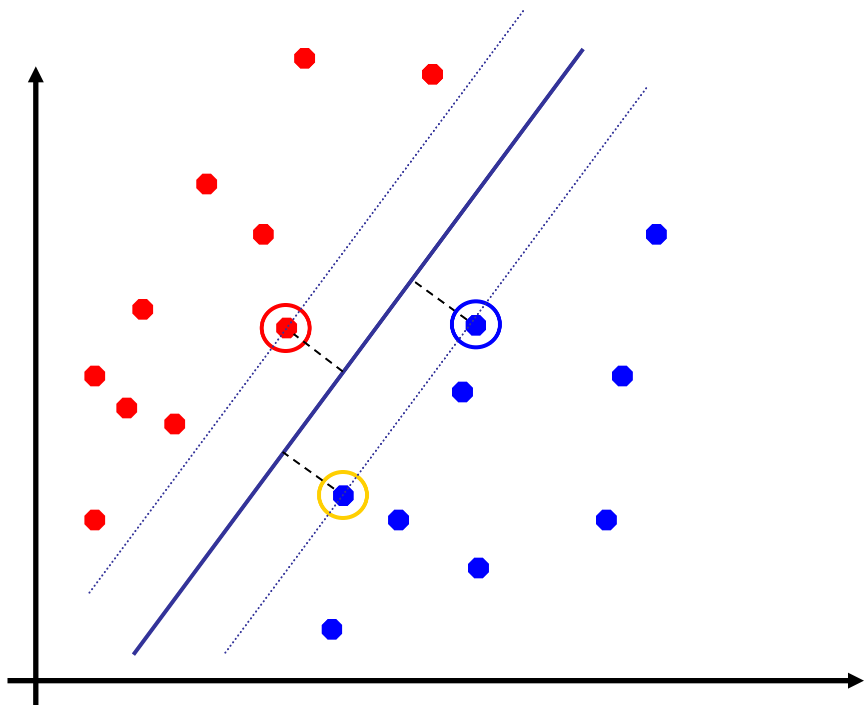
- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, *m*
  - Distance between the origin and the line $\mathbf{w}^t\mathbf{x}$=k is k/||$\mathbf{w}$||



$$m = \frac{2}{||\mathbf{w}||}$$

Maximum Margin Classification

- Maximizing the margin is good according to intuition and PAC theory.
- Implies that only support vectors matter; other training examples are ignorable.



Finding the Decision Boundary

- Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow$
- The $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i$ by solving the following constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

- This is a constrained optimization problem. Solving it requires some new tools
  - Feel free to ignore the following several slides; what is important is the constrained optimization problem above

Recap of Constrained Optimization

- Suppose we want to: minimize f(**x**) subject to g(**x**) = 0
- A necessary condition for $\mathbf{x}_0$ to be a solution:

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}}\Big(f(\mathbf{x}) + \alpha g(\mathbf{x})\Big)\Big|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0} \\ g(\mathbf{x}) = 0 \end{cases}$$

- α: the Lagrange multiplier
- For multiple constraints $g_i(\mathbf{x}) = 0$, i=1, …, m, we need a Lagrange multiplier α_i for each of the constraints

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}}\Big(f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i g_i(\mathbf{x})\Big)\Big|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0} \\ g_i(\mathbf{x}) = 0 \qquad \text{for } i = 1, \ldots, m \end{cases}$$

Recap of Constrained Optimization

- The case for inequality constraint $g_i(\mathbf{x}) \leq 0$ is similar, except that the Lagrange multiplier α_i should be positive
- If $\mathbf{x}_0$ is a solution to the constrained optimization problem

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \ldots, m$$

- There must exist $\alpha_i \geq 0$ for i=1, …, m such that $\mathbf{x}_0$ satisfy

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}} \left(f(\mathbf{x}) + \sum_i \alpha_i g_i(\mathbf{x}) \right) \Big|_{\mathbf{x}=jx_0} = \mathbf{0} \\ g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \ldots, m \end{cases}$$

- The function $f(\mathbf{x}) + \sum_i \alpha_i g_i(\mathbf{x})$ is also known as the Lagrangrian; we want to set its gradient to **0**

Back to the Original Problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } 1-y_i(\mathbf{w}^T\mathbf{x}_i+b) \leq 0 \qquad \text{for } i = 1, \ldots, n$$

- The Lagrangian is

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

  - Note that $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

- Setting the gradient of $\mathcal{L}$ w.r.t. **w** and b to zero, we have

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

The Dual Problem

- If we substitute $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$, we have $\mathcal{L}$

$$
\begin{aligned}
\mathcal{L} &= \frac{1}{2} \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \left(1 - y_i (\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b) \right) \\
&= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i y_i \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i - b \sum_{i=1}^{n} \alpha_i y_i \\
&= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i
\end{aligned}
$$

- Note that $\sum_{i=1}^{n} \alpha_i y_i = 0$

- This is a function of α_i only

The Dual Problem

- The new objective function is in terms of α_i only
- It is known as the dual problem: if we know **w**, we know all α_i; if we know all α_i, we know **w**
- The original problem is known as the primal problem
- The objective function of the dual problem needs to be maximized!
- The dual problem is therefore:

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Properties of α_i when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. b

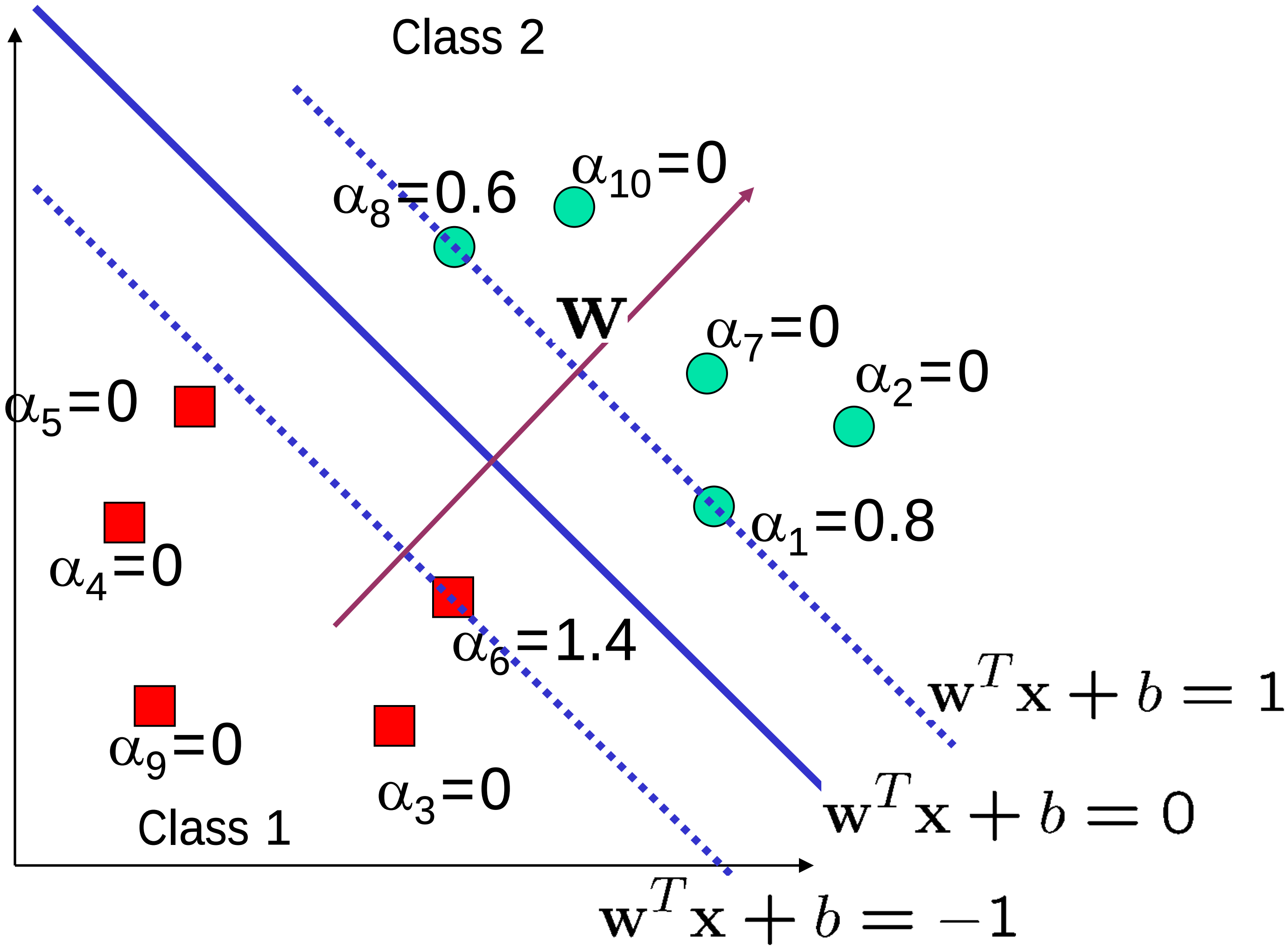
The Dual Problem

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- This is a quadratic programming (QP) problem
  - A global maximum of α_i can always be found
- **w** can be recovered by $$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

A Geometrical Interpretation



Characteristics of the Solution

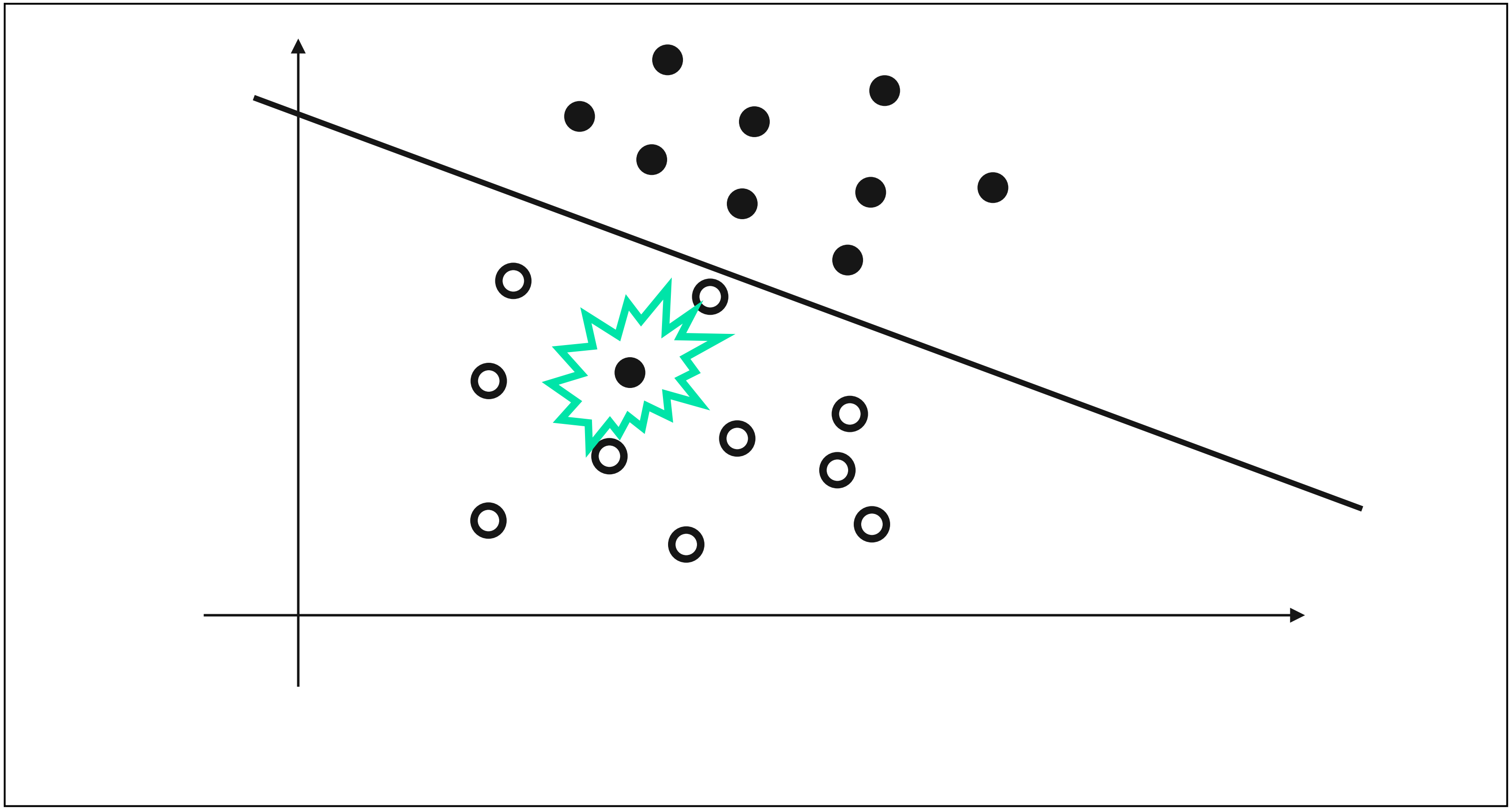
- Many of the α_i are zero
  - **w** is a linear combination of a small number of data points
  - This "sparse" representation can be viewed as data compression as in the construction of knn classifier
- $\mathbf{x}_i$ with non-zero α_i are called support vectors (SV)
  - The decision boundary is determined only by the SV
  - Let t_j (j=1, ..., s) be the indices of the s support vectors. We can write $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- For testing with a new data **z**
  - Compute $\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$ classify **z** as class 1 if the sum is positive, and class 2 otherwise
  - Note: **w** need not be formed explicitly

The Quadratic Programming Problem

- Many approaches have been proposed
  - Loqo, cplex, etc. (see http://www.numerical.rl.ac.uk/qp/qp.html)
- Most are “interior-point” methods
  - Start with an initial solution that can violate the constraints
  - Improve this solution by optimizing the objective function and/or reducing the amount of constraint violation
- For SVM, sequential minimal optimization (SMO) seems to be the most popular
  - A QP with two variables is trivial to solve
  - Each iteration of SMO picks a pair of (α_i,α_j) and solve the QP with these two variables; repeat until convergence
- In practice, we can just regard the QP solver as a “black-box” without bothering how it works

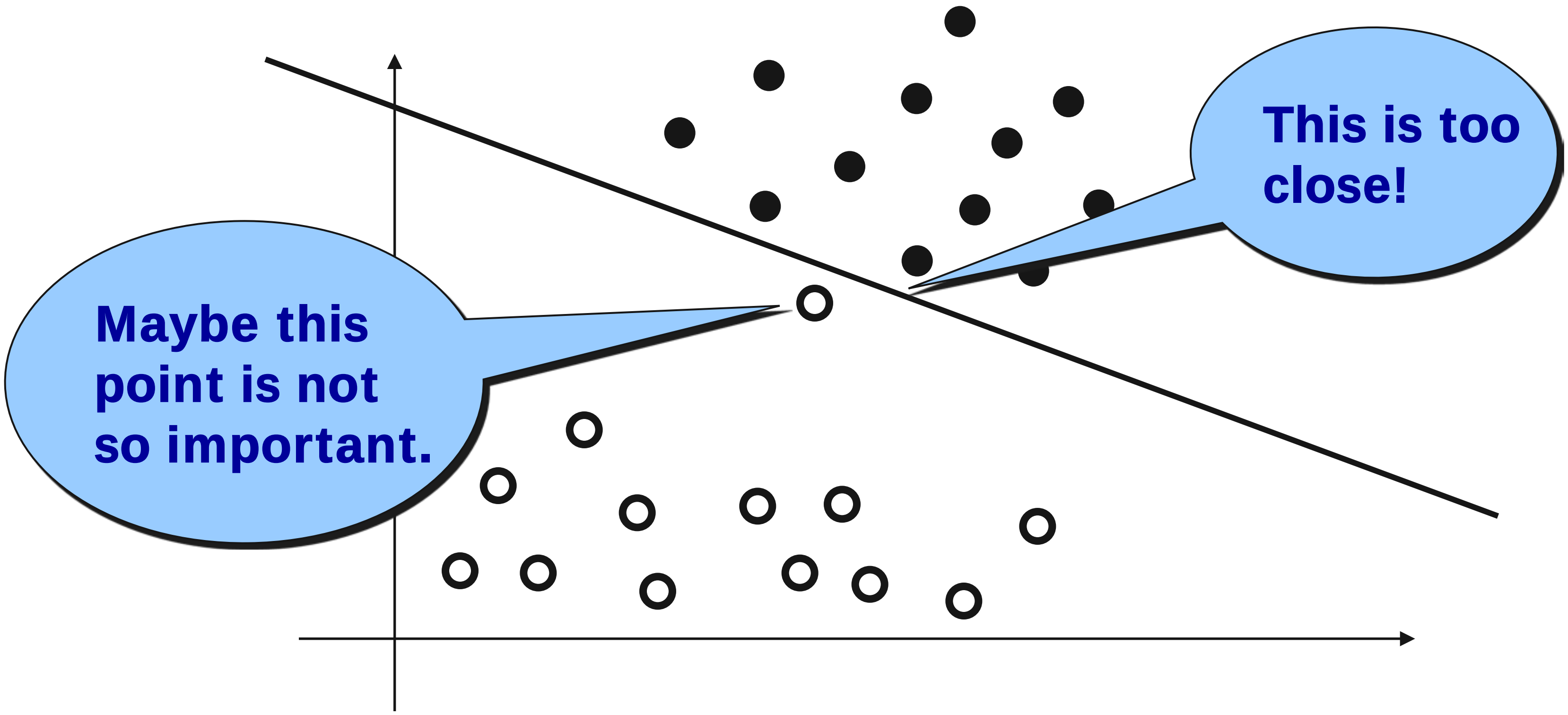
Non-Separable Sets

- **Sometimes, data sets are not linearly separable.**



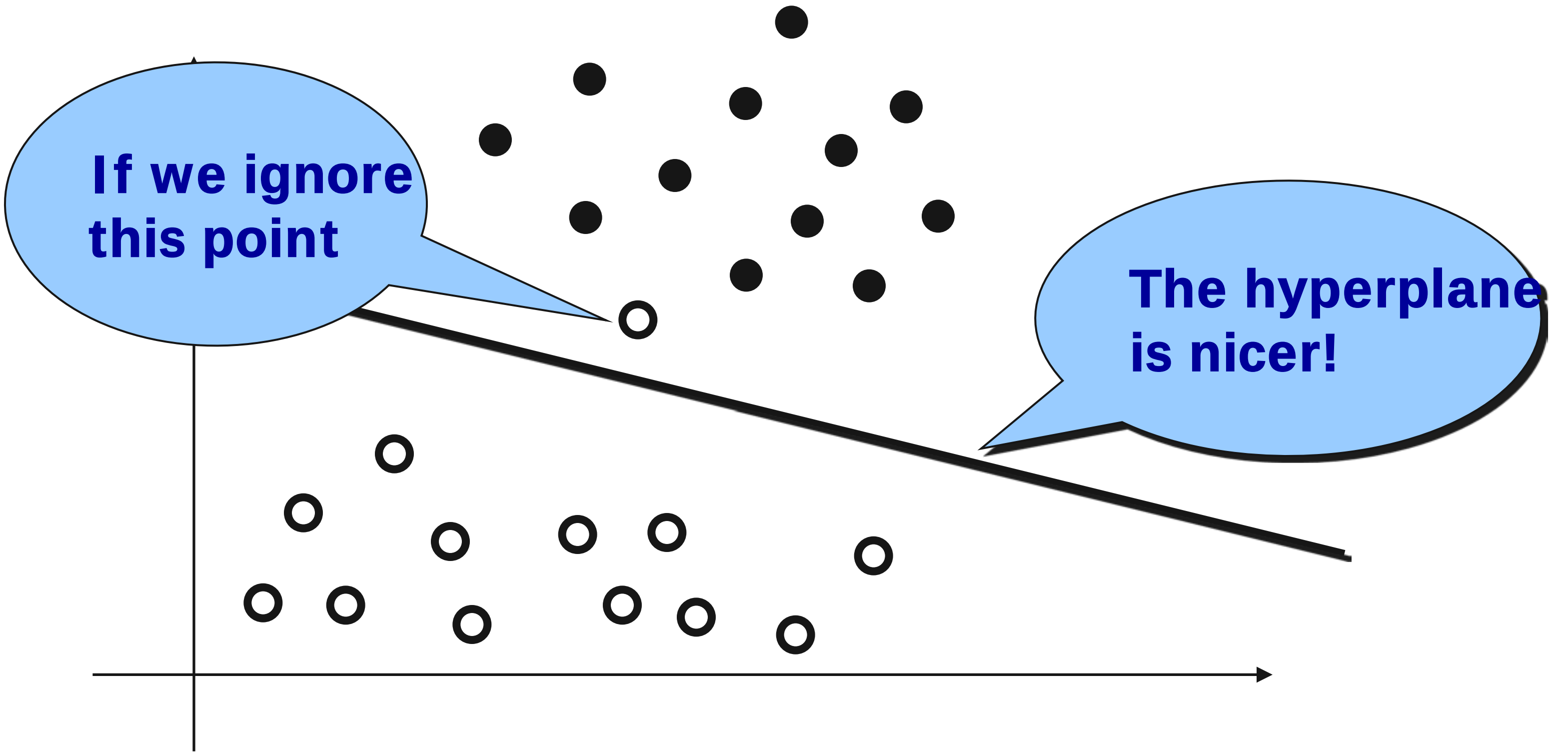
Non-Separable Sets

- Sometimes, we **do not** want to separate perfectly.



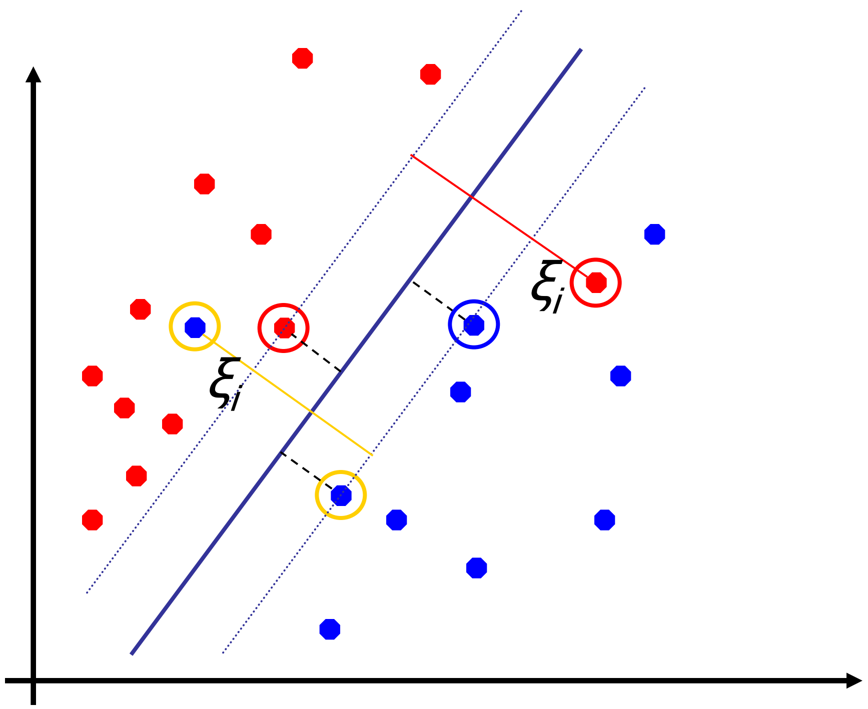
Non-Separable Sets

- Sometimes, we **do not** want to separate perfectly.



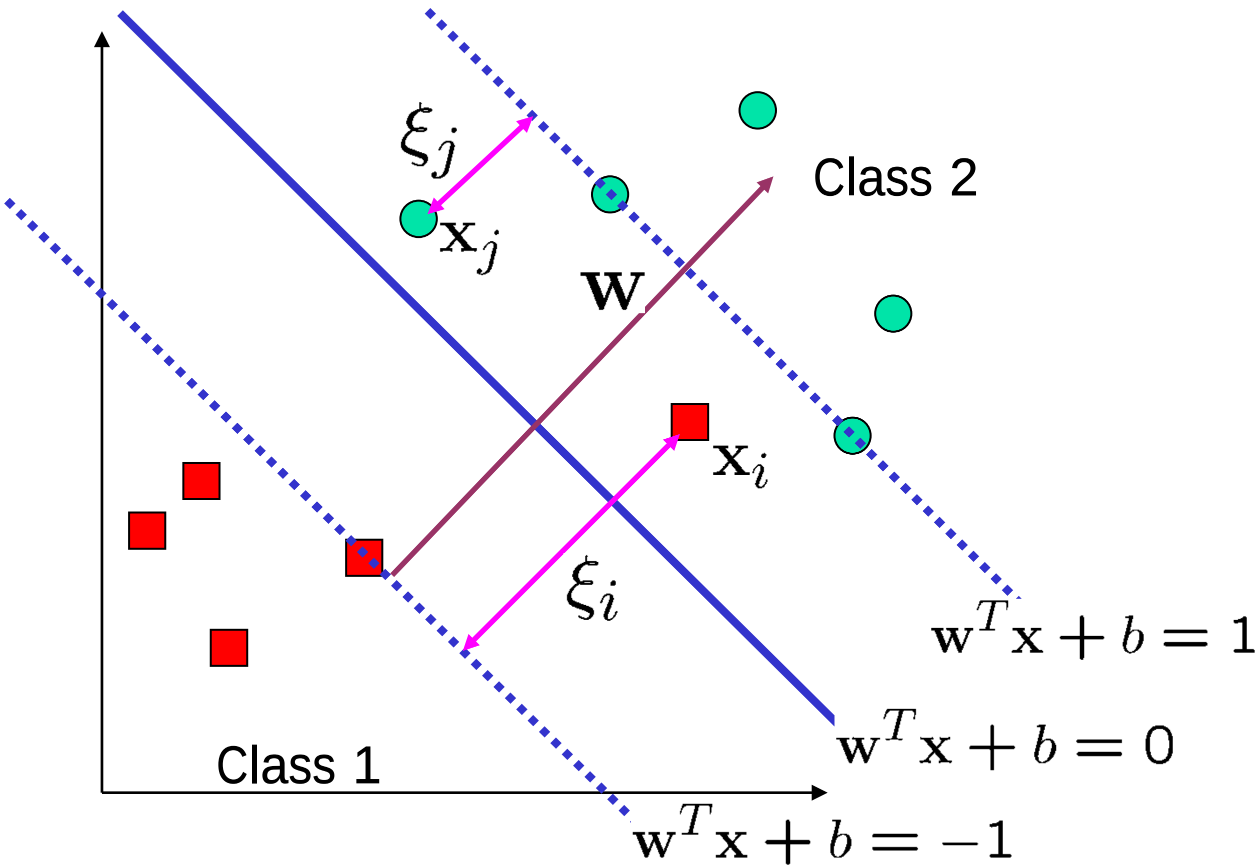
Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Non-linearly Separable Problems

- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}$+b
- ξ_i approximates the number of misclassified samples



Soft Margin Hyperplane

- If we minimize $\sum_i \xi_i$, ξ_i can be computed by

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

  - ξ_i are "slack variables" in optimization
  - Note that ξ_i=0 if there is no error for $\mathbf{x}_i$
  - ξ_i is an upper bound of the number of errors
- We want to minimize $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$
  - *C* : tradeoff parameter between error and margin
- The optimization problem becomes

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

The Optimization Problem

- The dual of this new constrained optimization problem is

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- **w** is recovered as $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$

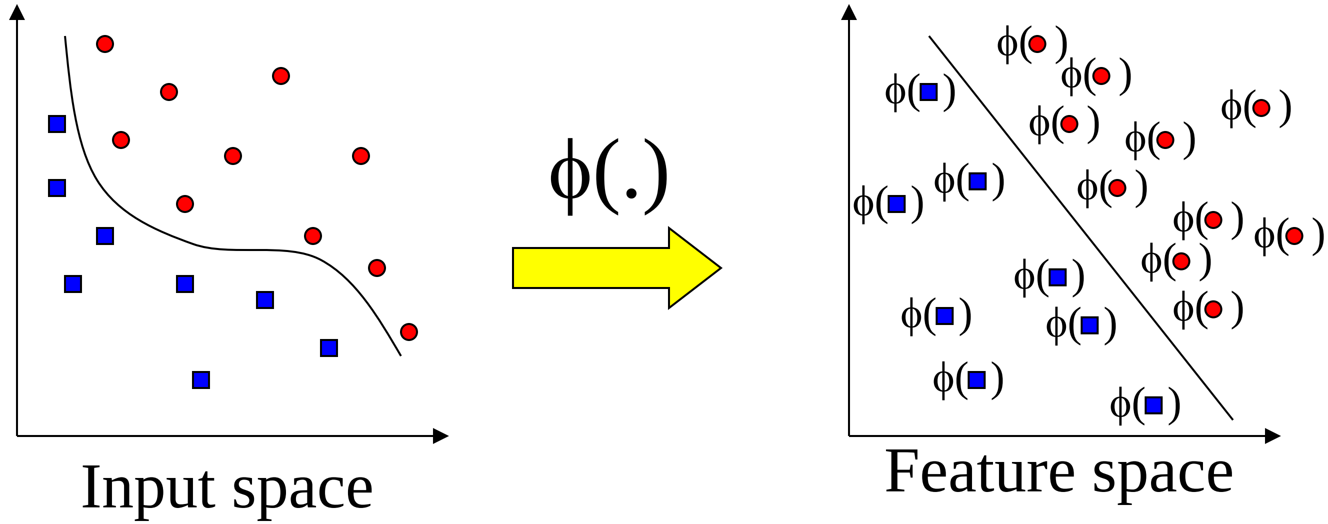
- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on α_i now
- Once again, a QP solver can be used to find α_i

SVM WITH KERNELS: LARGE-MARGIN NON-LINEAR CLASSIFIERS

Extension to Non-linear Decision Boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point $\mathbf{x}_i$ are located
  - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of x_1x_2 make the problem linearly separable

Transforming the Data (c.f. DHS Ch. 5)

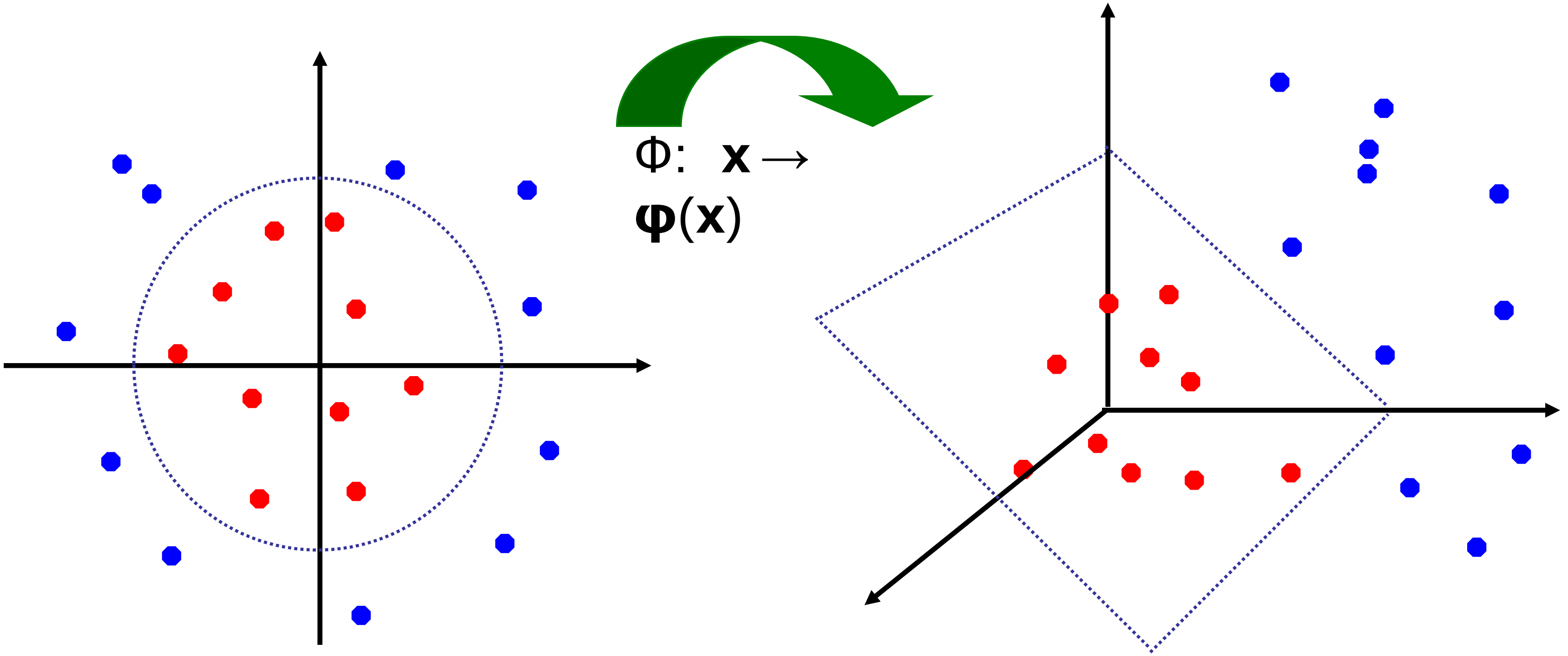


Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

Non-linear SVMs: Feature spaces

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



The Kernel Trick

- Recall the SVM optimization problem

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function *K* by

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

SVMs with kernels

- Training

$$\text{maximize}_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot K(\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subject to } \sum_{i=1}^{l} \alpha_i \cdot y_i = 0 \quad \text{and} \quad \forall i \;\; C \geq \alpha_i \geq 0$$

- Classification of $\mathbf{x}$:

$$h(\mathbf{x}) = sign\left(\sum_{i=1}^{l} \alpha_i \cdot y_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b \right)$$

An Example for ϕ(.) and K(.,.)

- Suppose ϕ(.) is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right)\right\rangle = (1 + x_1y_1 + x_2y_2)^2$$

- So, if we define the kernel function as follows, there is no need to carry out ϕ(.) explicitly

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- This use of kernel function to avoid carrying out ϕ(.) explicitly is known as the kernel trick

Kernel Functions

- Another view: kernel function, being an inner product, is really a similarity measure between the objects

Kernel Functions

- Any function $K(\mathbf{x},\mathbf{z})$ that creates a symmetric, positive definite matrix $K_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$ is a valid kernel (= an inner product in some space)

- Kernel (Gram) matrix:

$$\begin{pmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_2) & K(\mathbf{x}_1,\mathbf{x}_3) & \cdots & K(\mathbf{x}_1,\mathbf{x}_l) \\ K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & K(\mathbf{x}_2,\mathbf{x}_3) & & K(\mathbf{x}_2,\mathbf{x}_l) \\ \cdots & & & \cdots & \\ \cdots & & & \cdots & \\ K(\mathbf{x}_l,\mathbf{x}_1) & K(\mathbf{x}_l,\mathbf{x}_2) & K(\mathbf{x}_l,\mathbf{x}_3) & \cdots & K(\mathbf{x}_l,\mathbf{x}_l) \end{pmatrix}$$

Examples of Kernel Functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y}+1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x},\mathbf{y}) = \exp(-||\mathbf{x}-\mathbf{y}||^2/(2\sigma^2))$$

  - Closely related to radial basis function neural networks
  - The feature space is infinite-dimensional
- Sigmoid with parameter κ and θ

$$K(\mathbf{x},\mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y}+\theta)$$

  - It does not satisfy the Mercer condition on all κ and θ

Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

Original

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

With kernel function

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Modification Due to Kernel Function

- For testing, the new data **z** is classified as class 1 if $f \geq 0$, and as class 2 if $f < 0$

Original

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

More on Kernel Functions

- Since the training of SVM only requires the value of $K(\mathbf{x}_i, \mathbf{x}_j)$, there is no restriction of the form of $\mathbf{x}_i$ and $\mathbf{x}_j$
  - $\mathbf{x}_i$ can be a sequence or a tree, instead of a feature vector
- $K(\mathbf{x}_i, \mathbf{x}_j)$ is just a similarity measure comparing $\mathbf{x}_i$ and $\mathbf{x}_j$
- For a test object **z**, the discriminat function essentially is a weighted sum of the similarity between z and a pre-selected set of objects (the support vectors)

$$f(\mathbf{z}) = \sum_{\mathbf{x}_i \in \mathcal{S}} \alpha_i y_i K(\mathbf{z}, \mathbf{x}_i) + b$$

$\mathcal{S}$: the set of support vectors

More on Kernel Functions

- Not all similarity measure can be used as kernel function, however
  - The kernel function needs to satisfy the Mercer function, i.e., the function is “positive-definite”
  - This implies that the *n* by *n* kernel matrix, in which the (i,j)-th entry is the $K(\mathbf{x}_i, \mathbf{x}_j)$, is always positive definite
  - This also means that the QP is convex and can be solved in polynomial time

Example

- Suppose we have 5 1D data points
  - $x_1=1$, $x_2=2$, $x_3=4$, $x_4=5$, $x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ $y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i (i=1, …, 5) by

$$\text{max.} \quad \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Example

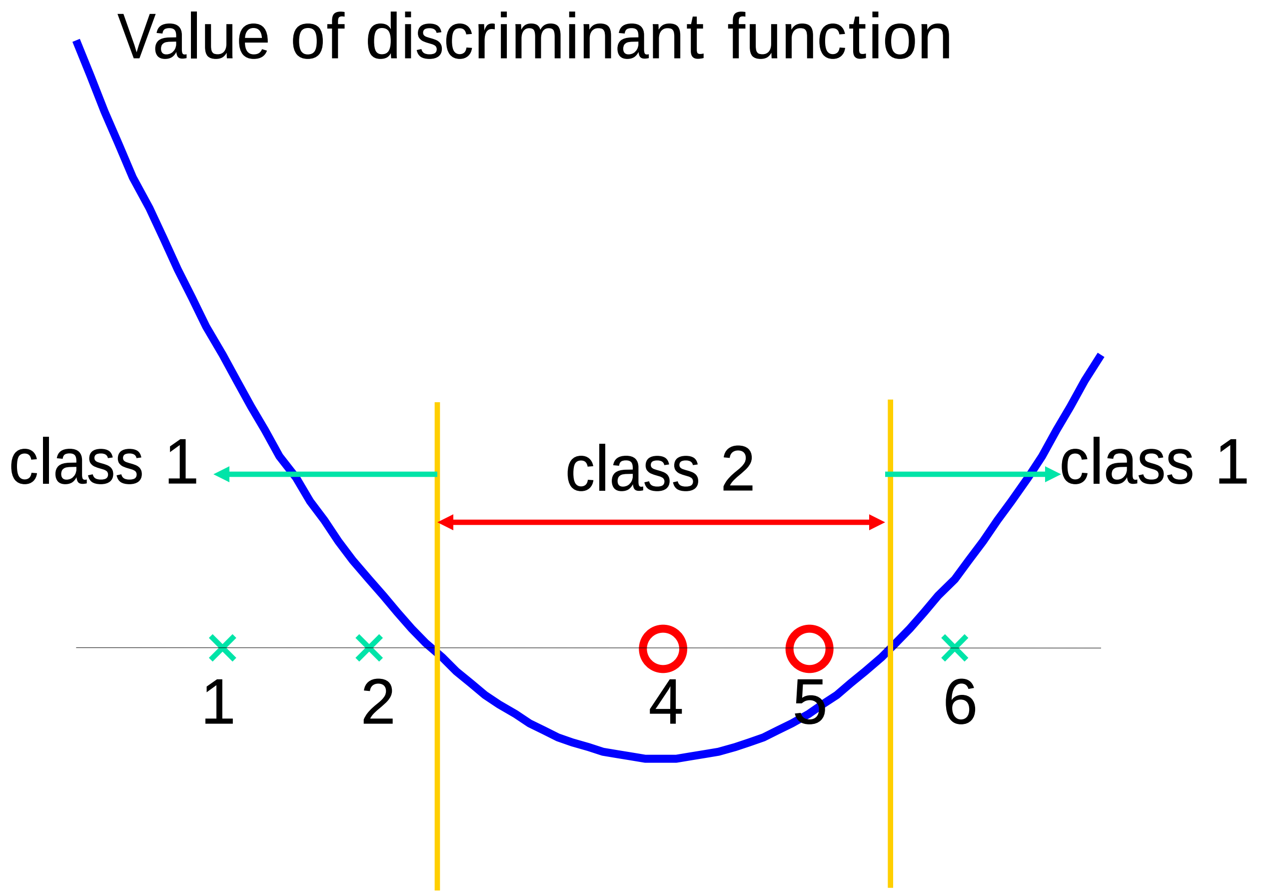
- By using a QP solver, we get
  - $\alpha_1=0$, $\alpha_2=2.5$, $\alpha_3=0$, $\alpha_4=7.333$, $\alpha_5=4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2=2, x_4=5, x_5=6\}$
- The discriminant function is

$$
\begin{aligned}
& f(z) \\
&= 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + \overset{\alpha_5}{4.833}\overset{y_5}{(1)}\overset{K(z,x_5)}{(6z+1)^2} + b \\
&= 0.6667z^2 - 5.333z + b
\end{aligned}
$$

- *b* is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2 and x_5 lie on the line $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = 1$n the line
- All three give b=9 $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = -1$

$$\Rightarrow f(z) = 0.6667z^2 - 5.333z + 9$$

Example


Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarizes all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)
- There is even research to estimate the kernel matrix from available information

- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try
- Note that SVM with RBF kernel is closely related to RBF neural networks, with the centers of the radial basis functions automatically chosen for SVM

Other Aspects of SVM

- How to use SVM for multi-class classification?
  - One can change the QP formulation to become multi-class
  - More often, multiple binary classifiers are combined
    - See DHS 5.2.2 for some discussion
  - One can train multiple one-versus-all classifiers, or combine multiple pairwise classifiers “intelligently”
- How to interpret the SVM discriminant function value as probability?
  - By performing logistic regression on the SVM output of a set of data (validation set) that is not used for training
- Some SVM software (like libsvm) have these features built-in

Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Summary: Steps for Classification

- Prepare the pattern matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
    - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

Strengths and Weaknesses of SVM

- Strengths
  - Training is relatively easy
    - No local optimal, unlike in neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
- Weaknesses
  - Need to choose a “good” kernel function.

Other Types of Kernel Methods

- A lesson learnt in SVM: a linear algorithm in the feature space is equivalent to a non-linear algorithm in the input space
- Standard linear algorithms can be generalized to its non-linear version by going to the feature space
  - Kernel principal component analysis, kernel independent component analysis, kernel canonical correlation analysis, kernel k-means, 1-class SVM are some examples

Conclusion

- SVM is a useful alternative to neural networks
- Two key concepts of SVM: maximize the margin and the kernel trick
- Many SVM implementations are available on the web for you to try on your data set!

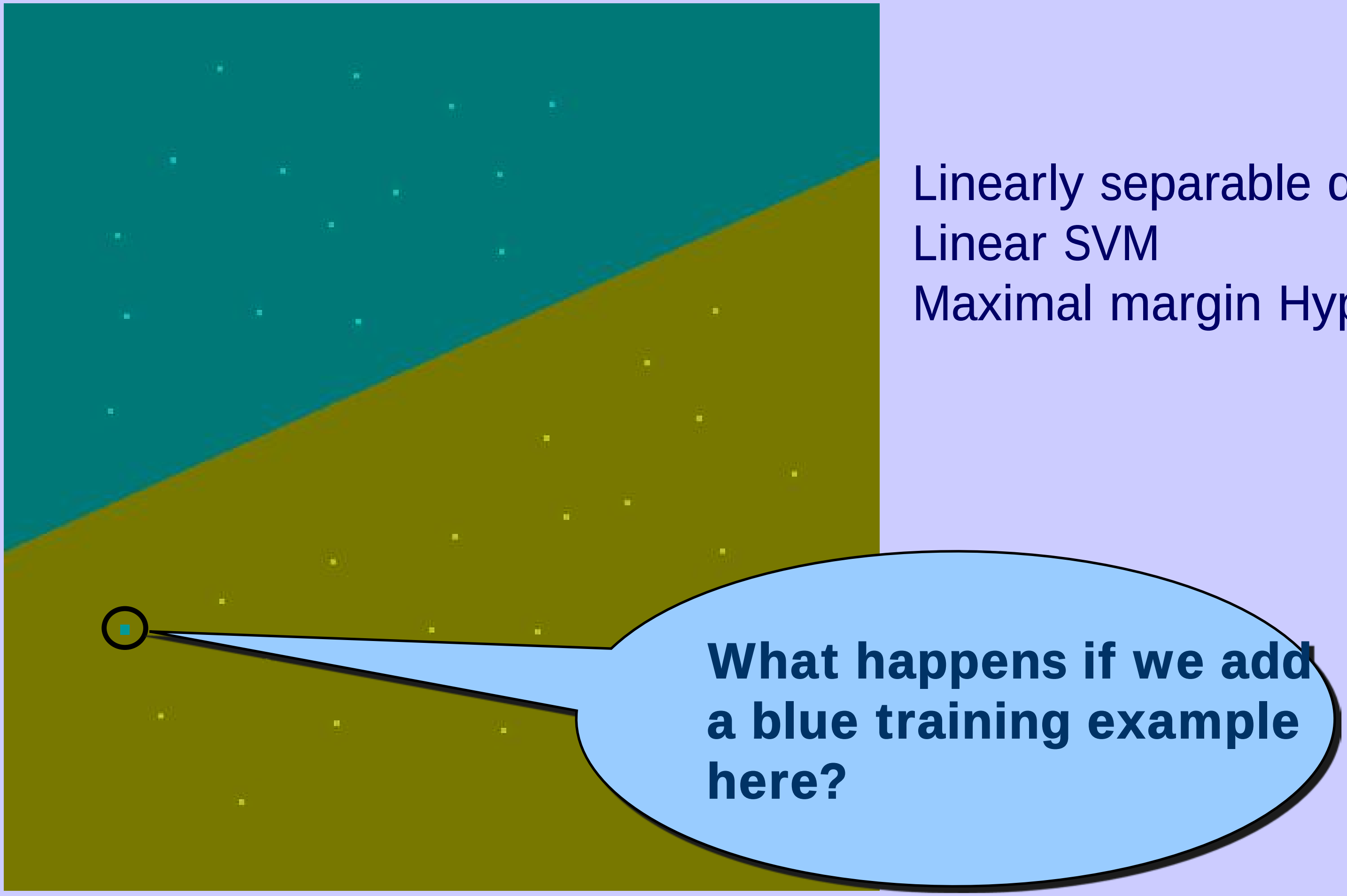
Toy Examples

- All examples have been run with the 2D graphic interface of SVMLIB (Chang and Lin, National University of Taiwan)

  "**LIBSVM** is an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, un-SVR) and distribution estimation (one-class SVM). It supports multi-class classification. The basic algorithm is a simplification of both SMO by Platt and SVMLight by Joachims. It is also a simplification of the modification 2 of SMO by Keerthy et al. Our goal is to help users from other fields to easily use SVM as a tool. **LIBSVM** provides a simple interface where users can easily link it with their own programs…"

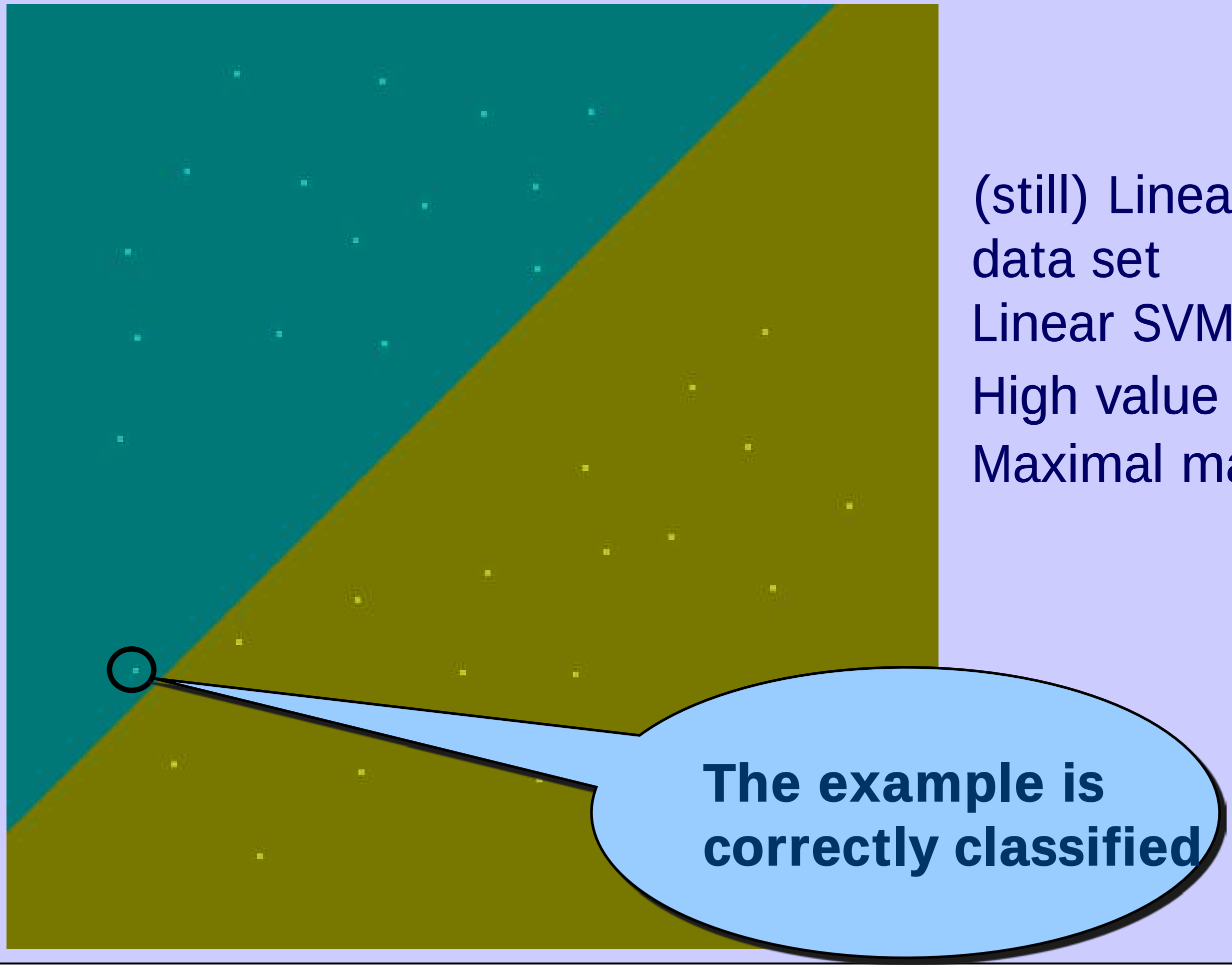
- Available from: **www.csie.ntu.edu.tw/~cjlin/libsvm** (it icludes a Web integrated demo tool)

Toy Examples (I)



Linearly separable data set
Linear SVM
Maximal margin Hyperplane

Toy Examples (I)



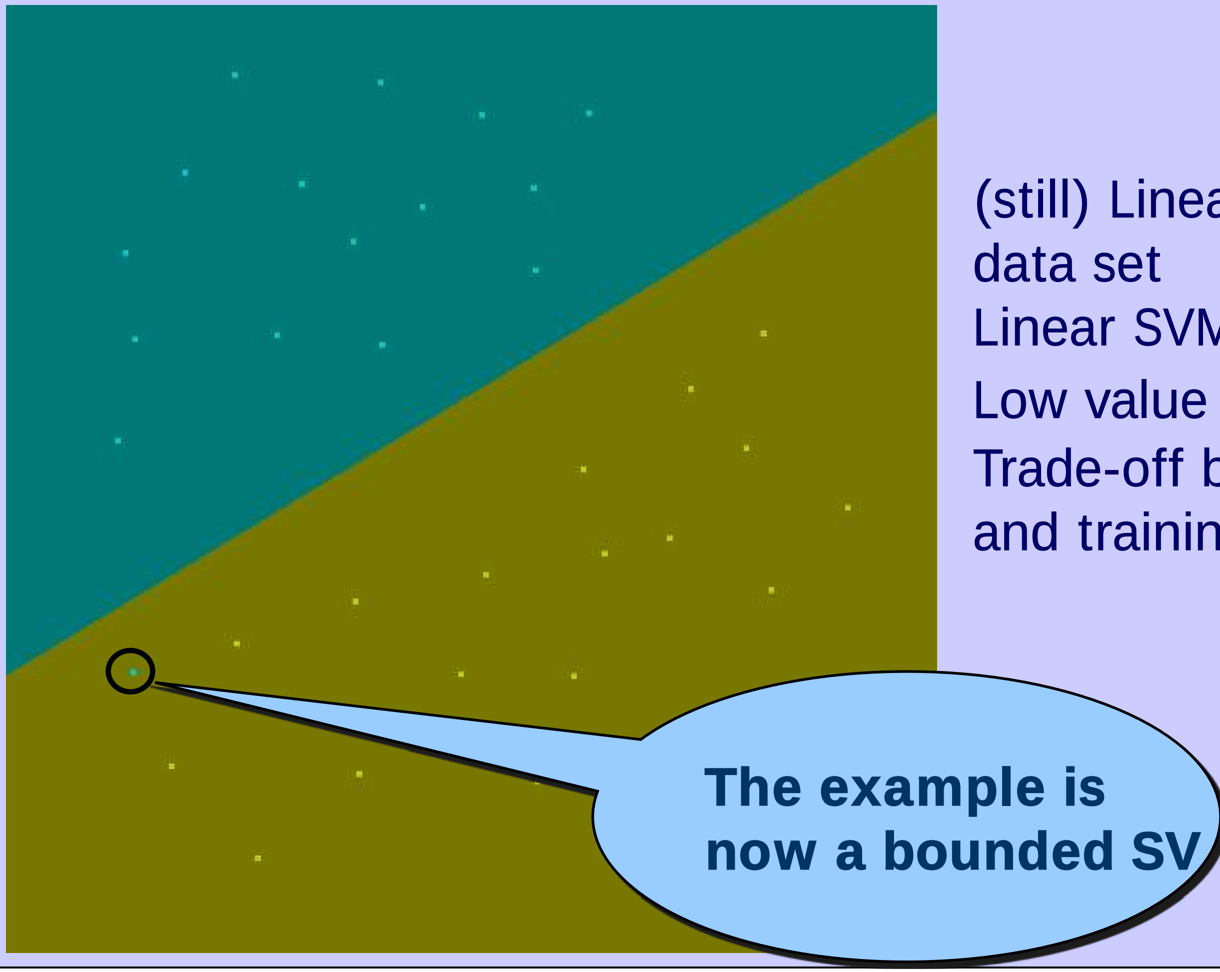
(still) Linearly separable data set

Linear SVM

High value of C parameter

Maximal margin Hyperplane

Toy Examples (I)



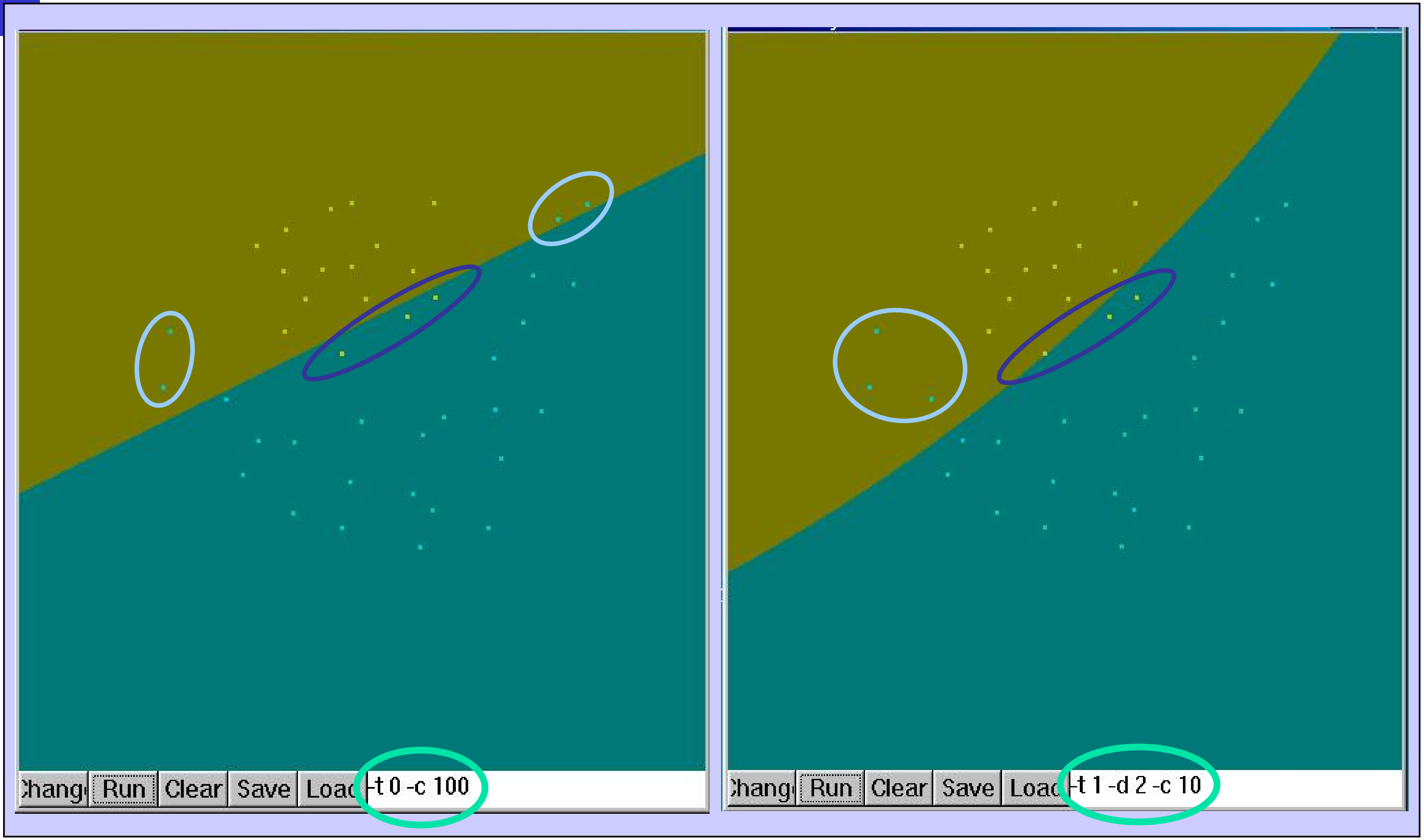
(still) Linearly separable data set

Linear SVM

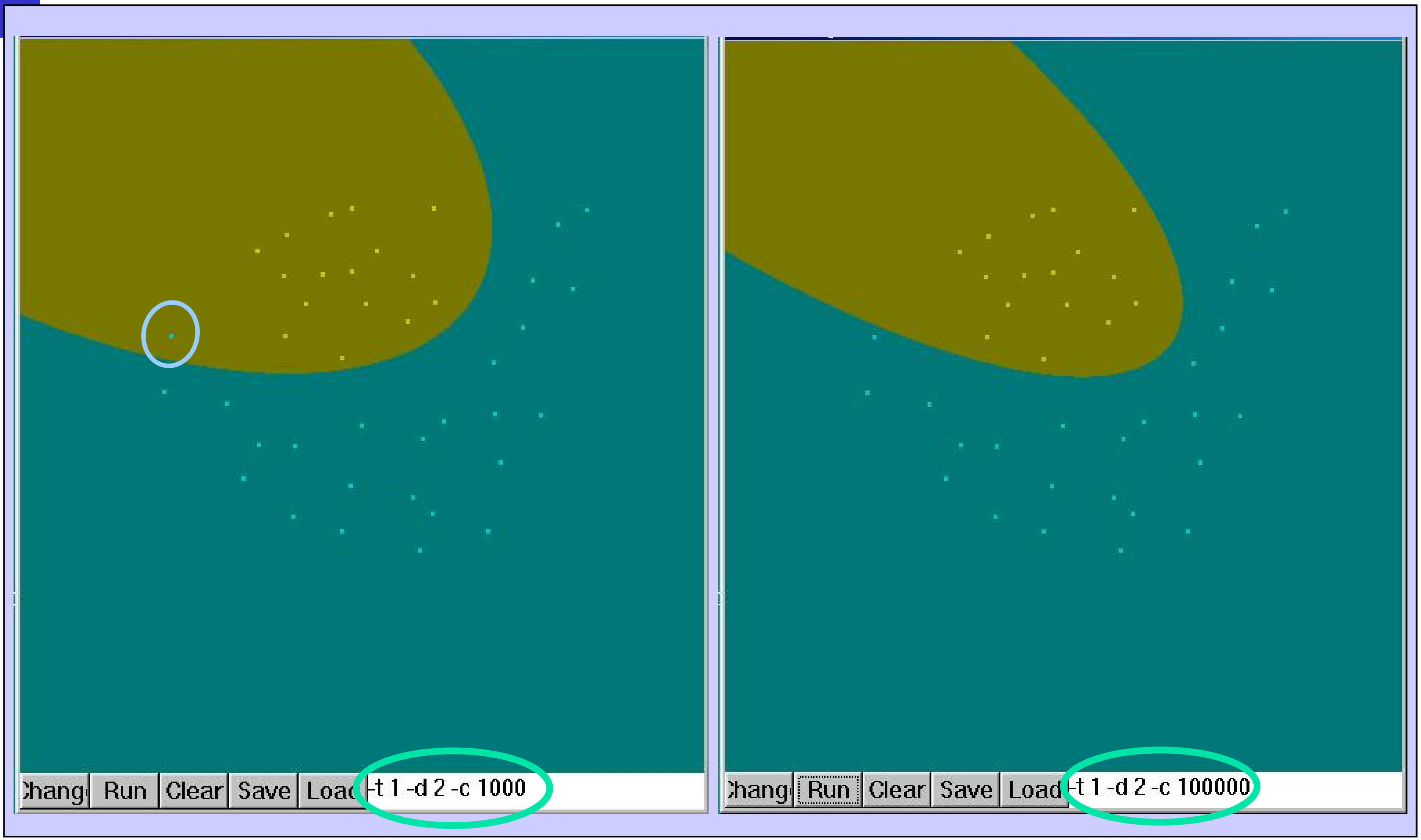
Low value of C parameter

Trade-off between: margin and training error

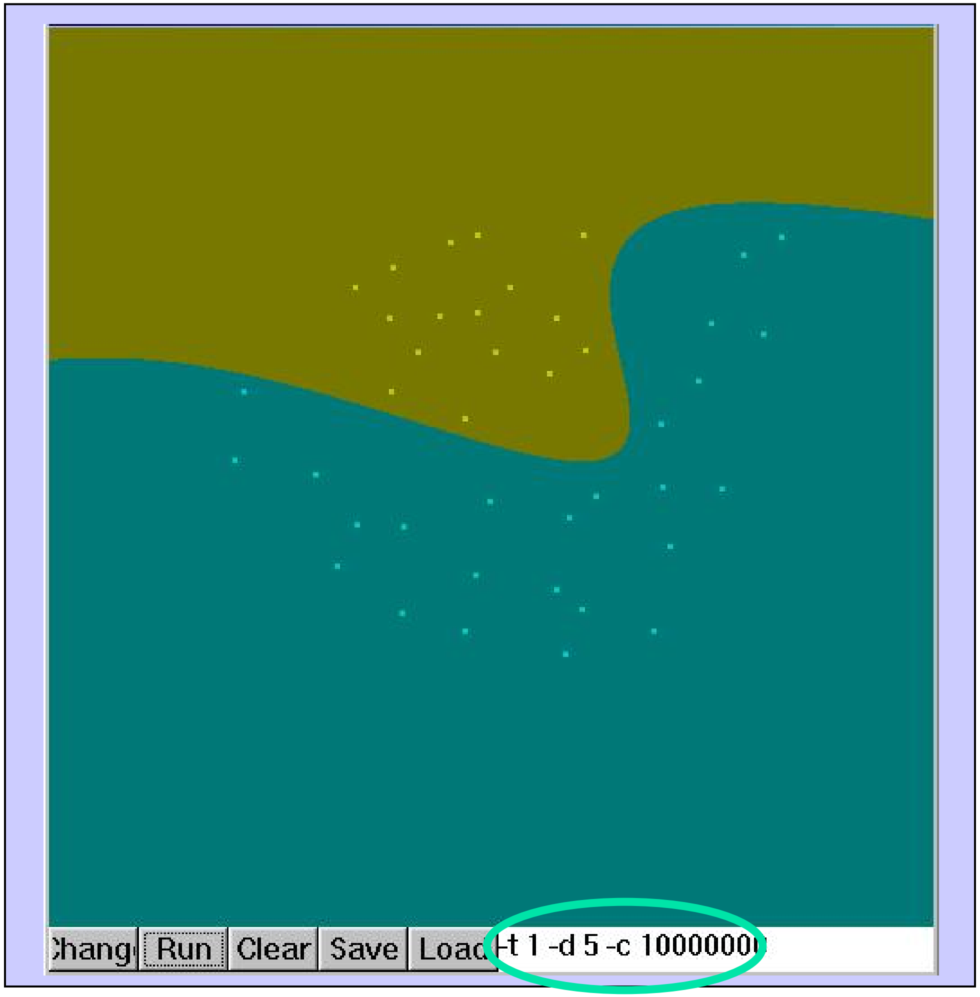
Toy Examples (I)



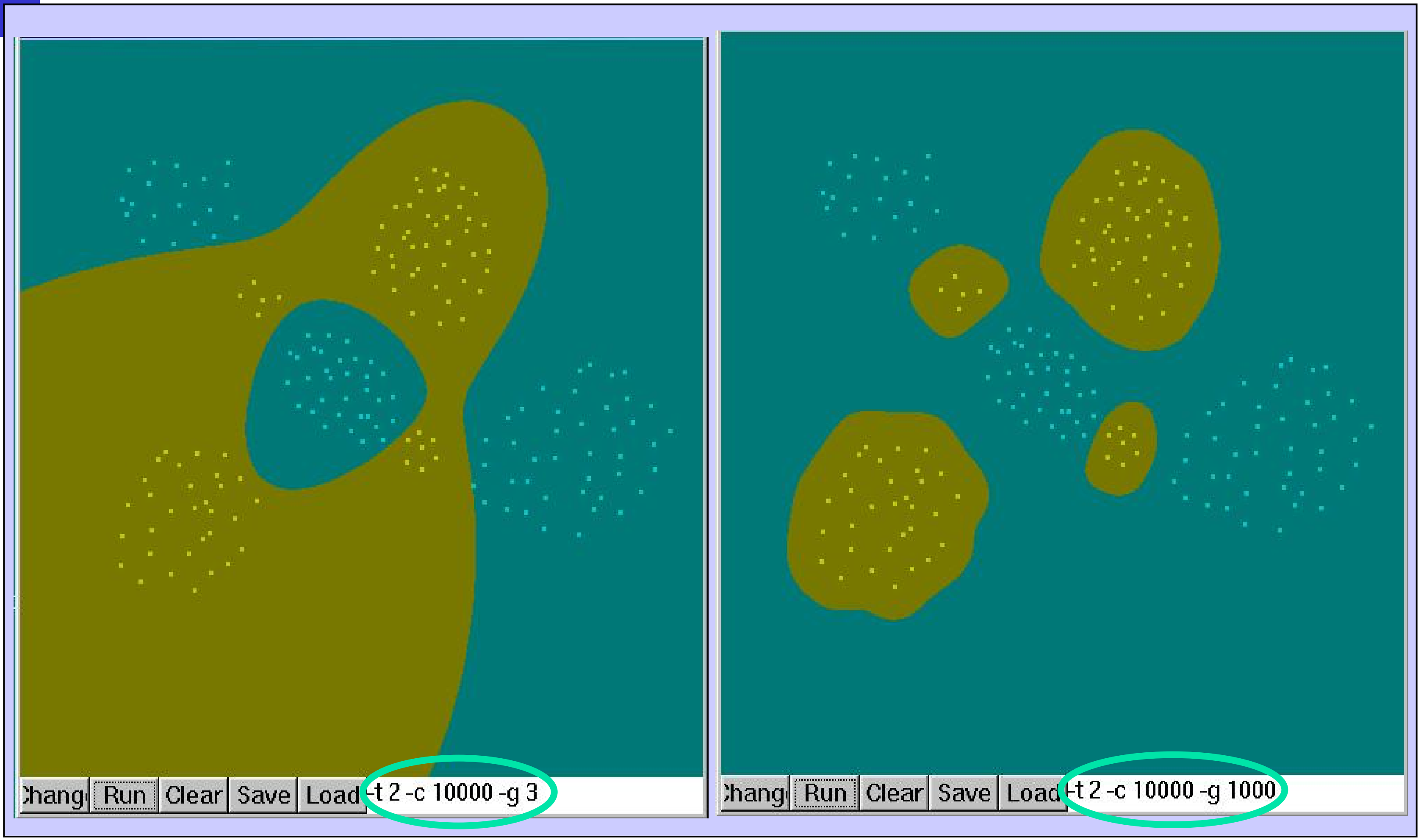
Toy Examples (I)



Toy Examples (I)



Toy Examples (I)



Resources

- http://www.kernel-machines.org/
- http://www.support-vector.net/
- http://www.support-vector.net/icml-tutorial.pdf
- http://www.kernel-machines.org/papers/tutorial-nips.ps.gz
- http://www.clopinet.com/isabelle/Projects/SVM/applist.html

APPLICATIONS

SVM applications

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97], principal component analysis [Schölkopf *et al.* '99], etc.

The problem

Text Categorization (**TC**) is the problem of classifying free text documents into a set of predefined categories or classes

Comprehensive and nice survey in **(Sebastiani, 2001)**

- **TC** properties:
  - **Multiclass** & **multi-label** classification problem
  - Very **high dimensionality**
  - Extremely **Sparse** instance vectors
  - **Dense** concept vector

Applications of TC

- **Document filtering**:
  - news agencies, spam e-mail, netnews, etc.
- **Document routing**:
  - news agencies, message (e-mail) classif., IR, etc.
- **(Re)organizing a document collection**:
  - textual DB's, Internet, etc.
- **Hierarchical categorization**
  - Web pages
- **Semantic disambiguation tasks**

Example

JAPAN BUYS MODEST AMOUNT OF DOLLARS
24-MAR-1987, 20:06:00.50

The Bank of Japan bought a modest amount of dollars this morning, possibly around 200 to 300 mln, dealers said.
One dealer said the central bank bought about 200 mln dlrs through brokers and the rest through banks. The buying began when the dollar was at about 149.60 yen, and helped drive the U.S. Currency up to around 150, he said.
Another said the central bank seemed to be trying to push the dollar up above 150 yen. But heavy selling at around that level quickly pushed the dollar back down towards 149 yen, dealers said. REUTER

CLASSES: **money-fx dollar**

Document representation

Training documents are represented as a set of features plus a set of associated labels

- Which features?
  - Bag of words: each word occurring in a document. Stemming and Stop-word list can be used
  - Linguistic phrases (usually extracted using IE methods)
  - First-order relations
- Feature values can be:
  - Binary: appears or not
  - A function of the number of occurrences
  - **TFIDF values**

A simple TF-IDF "kernel"

$$tfidf(t,x) = \#(t,x) \cdot \log \frac{|T|}{\#_T(t)}$$

$$w_{i,x} = \phi_i(x) = \frac{tfidf(t_i,x)}{\sqrt{\sum_{y \in T} (tfidf(t_i,y))^2}}$$

- The more often a term occurs in a document the more it is representative of its content
- The more documents the term occurs in, the less discriminating it is

Example

CLASSES: **{money-fx,dollar}**
FEATURES:
###6
###.## 1
##:##:##.## 1
##-mar-#### 1
amount 3
another 1
back 1
bank 3
banks 1
...
u.s. 1
was 1
yen 3

Evaluation Measures

- Precision

$$P = \frac{\text{categories predicted and correct}}{\text{total categories predicted}}$$

- Recall

$$R = \frac{\text{categories predicted and correct}}{\text{total categories correct}}$$

- BEP: The point where precision and recall are equal

- $F_\beta(R, P)$; [Usually $\beta = 1$]

$$F_\beta(R, P) = \frac{(\beta^2+1)PR}{\beta^2 P+R}$$

- One-Error, Coverage, Average Precision,
  11-point interpolated Average Precision

Te

			#1	#2	#3	#4	#5
		# of documents	21,450	14,347	13,272	12,902	12,902
		# of training documents	14,704	10,667	9,610	9,603	9,603
		# of test documents	6,746	3,680	3,662	3,299	3,299
		# of categories	135	93	92	90	10
System	**Type**	**Results reported by**					
Word	(non-learning)	[Yang 1999]	.150	.310	.290		
	probabilistic	[Dumais et al. 1998]				.752	.815
	probabilistic	[Joachims 1998]					.720
	probabilistic	[Lam et al. 1997]	.443 (MF_1)				
PropBayes	probabilistic	[Lewis 1992a]	.650				
Bim	probabilistic	[Li and Yamanishi 1999]				.747	
	probabilistic	[Li and Yamanishi 1999]				.773	
Nb	probabilistic	[Yang and Liu 1999]				.795	
	decision trees	[Dumais et al. 1998]					.884
C4.5	decision trees	[Joachims 1998]					.794
Ind	decision trees	[Lewis and Ringuette 1994]	.670				
Swap-1	decision rules	[Apté et al. 1994]		.805			
Ripper	decision rules	[Cohen and Singer 1999]	.683	.811		.820	
SleepingExperts	decision rules	[Cohen and Singer 1999]	**.753**	.759		.827	
Dl-Esc	decision rules	[Li and Yamanishi 1999]				.820	
Charade	decision rules	[Moulinier and Ganascia 1996]		.738			
Charade	decision rules	[Moulinier et al. 1996]		.783 (F_1)			
Llsf	regression	[Yang 1999]		.855	.810		
Llsf	regression	[Yang and Liu 1999]				.849	
BalancedWinnow	on-line linear	[Dagan et al. 1997]	.747 (M)	.833 (M)			
Widrow-Hoff	on-line linear	[Lam and Ho 1998]				.822	
Rocchio	batch linear	[Cohen and Singer 1999]	.660	.748		.776	
FindSim	batch linear	[Dumais et al. 1998]				.617	.646
Rocchio	batch linear	[Joachims 1998]					.799
Rocchio	batch linear	[Lam and Ho 1998]				.781	
Rocchio	batch linear	[Li and Yamanishi 1999]				.625	
Classi	neural network	[Ng et al. 1997]		.802			
Nnet	neural network	[Yang and Liu 1999]				.838	
	neural network	[Wiener et al. 1995]			**.820**		
Gis-W	example-based	[Lam and Ho 1998]				.860	
k-NN	example-based	[Joachims 1998]					.823
k-NN	example-based	[Lam and Ho 1998]				.820	
k-NN	example-based	[Yang 1999]	.690	.852	**.820**		
k-NN	example-based	[Yang and Liu 1999]				.856	
	SVM	[Dumais et al. 1998]				.870	**.920**
SvmLight	SVM	[Joachims 1998]					.864
SvmLight	SVM	[Li and Yamanishi 1999]				.841	
SvmLight	SVM	[Yang and Liu 1999]				.859	
AdaBoost.MH	committee	[Schapire and Singer 2000]		**.860**			
	committee	[Weiss et al. 1999]				**.878**	
	Bayesian net	[Dumais et al. 1998]				.800	.850
	Bayesian net	[Lam et al. 1997]	.542 (MF_1)				

(Sebastiani, 2001)

TC using SVMs

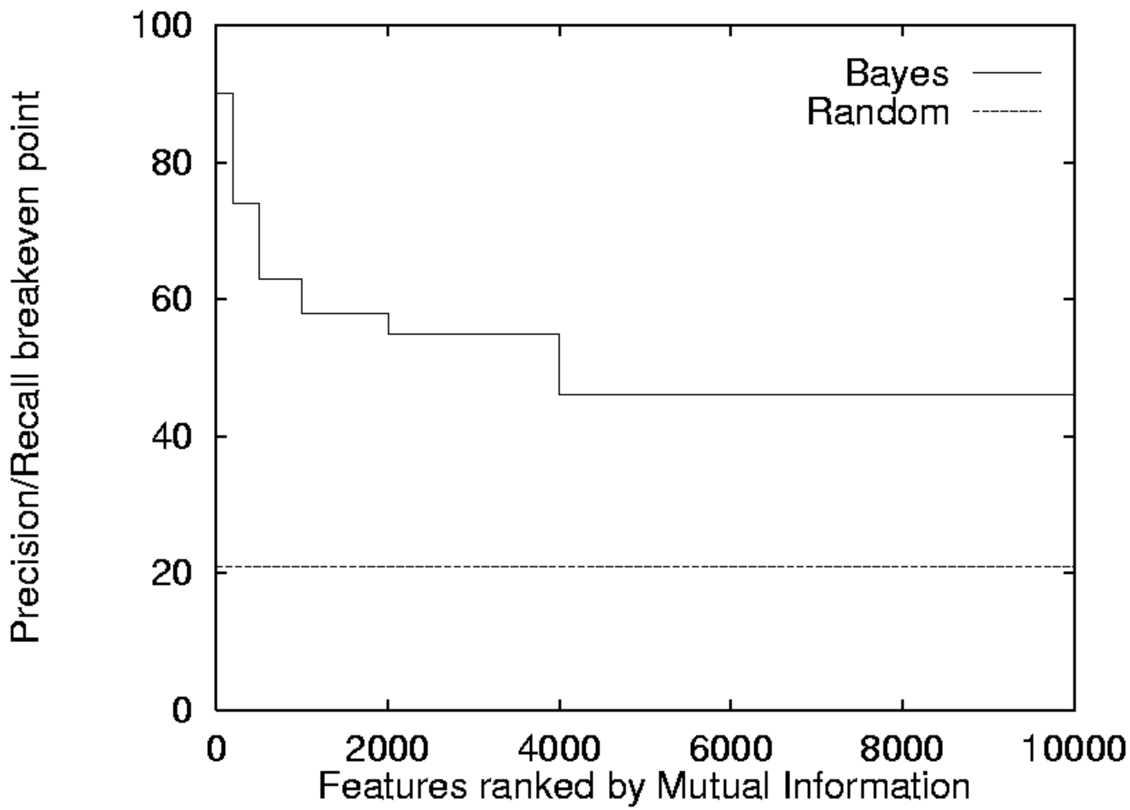
(Dumais, et al., 1998) *SIKM'98*

- Empirical evaluation
  - **Reuters-21578** corpus
  - Agressive attribute filtering using Mutual Information: 300 binary terms
  - Linear SVM's
  - Significantly outperforms a number of systems in a comparative experiment. BEP on Reuters = **87.0**
  - Also impressive training and classification speed (0.26 CPU seconds to train each category)

Why they are appropriate?

- (Joachims, 1998)
  - SVMs are tolerant to overfitting in high dimensional input spaces
  - Few irrelevant features (dense concept vector)
  - Document vectors are sparse (Kivinen & Warmuth, 1995)
  - Most text categorization problems are linearly separable.
  - SVMs are relatively insensitive to the relative number of training instances of each class

Few irrelevant features



(Joachims, 1998)

TRANSDUCTION

Transductive SVMs

- **Transductive** instead of inductive (Vapnik 98)
- TSVMs take into account a particular test set and try to minimize misclassifications of just those particular examples
- Formal setting:

$$S_{train} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$$

$$S_{test} = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \ \ldots, \mathbf{x}_k^*\} \ \text{(normally } k >> n)$$

Goal of the transductive learner L :

find a function $h_L = L(S_{train}, S_{test})$ so that the expected number of erroneous predictions on the test examples is minimized

(Joachims, 1999)

Transductive SVMs

- Appropriate tasks: all have in common that there is little training data, but a very large training set, e.g. text classification tasks (Joachims, 99):
  - **Relevance Feedback**
  - **Netnews Filtering**
  - **Reorganizing a collection of documents**

Transductive SVMs

Why should TSVMs perform better than SVMs?

In the field of information retrieval it is well known that words in natural language occur in strong co-ocurrence patterns

	Class	nuclear	physics	atom	parsley	basil	salt	and
Train1	A	1	-	-	-	-	-	1
Train2	B	-	-	-	-	1	1	1
Test1	?	1	1	1	-	-	-	1
Test2	?	-	-	1	-	-	-	1
Test3	?	-	-	-	1	1	-	1
Test4	?	-	-	-	1	-	1	1

(Joachims, 1999)

Transducive SVMs

Why should TSVMs perform better than SVMs?

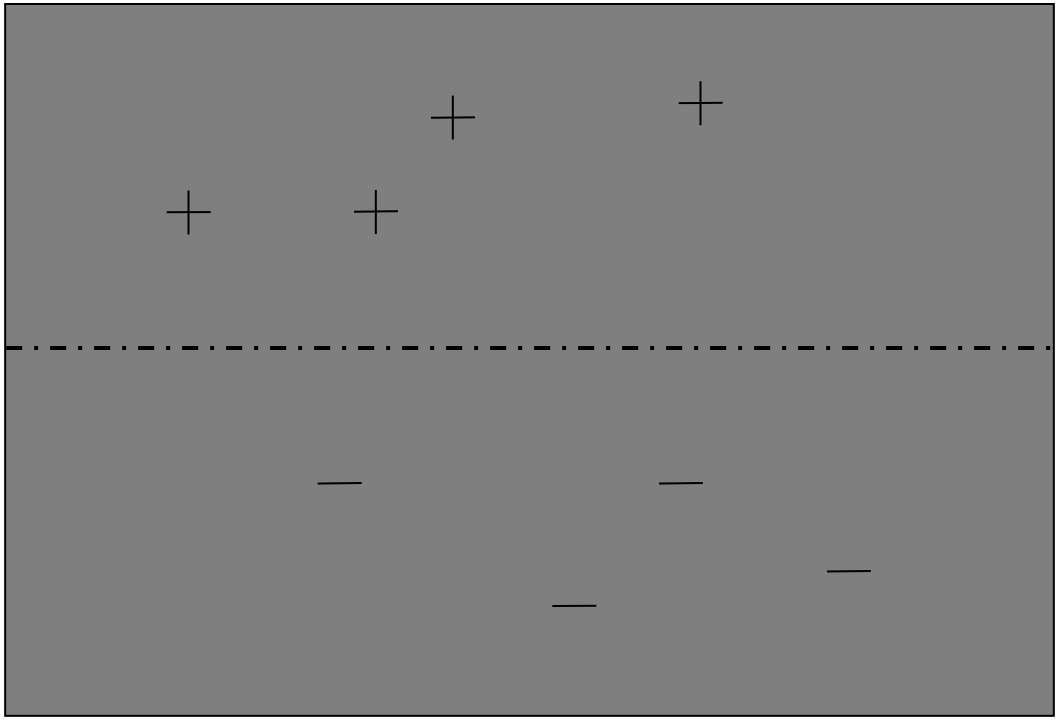
In the field of information retrieval it is well known that words in natural language occur in strong co-ocurrence patterns

	Class	nuclear	physics	atom	parsley	basil	salt	and
Train1	A	**1**	-	-	-	-	-	1
Train2	B	-	-	-	-	1	1	1
Test1	**A**	**1**	1	**1**	-	-	-	1
Test2	**A**	-	-	**1**	-	-	-	1
Test3	**B**	-	-	-	1	1	-	1
Test4	**B**	-	-	-	1	-	1	1

(Joachims, 1999)

Transductive SVMs

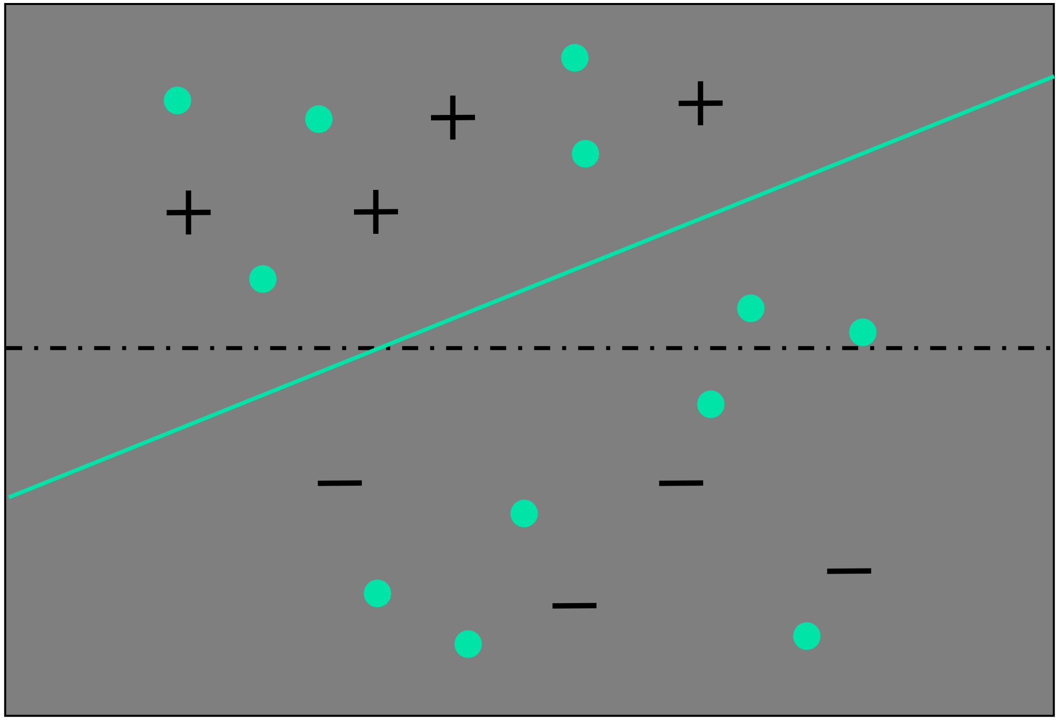
How does the transductive approach work?



(Joachims, 1999)

Transductive SVMs

How does the transductive approach work?



(Joachims, 1999)

Transductive SVMs

- Empirical evaluation (Joachims 99)
  - Comparison between Naive Bayes, SVMs and TSVMs
  - Datasets:
    - **Reuters-21578** corpus (News agency)
    - **WebKB** collection (Documents: Web pages from a university)
    - **Ohsumed** corpus (Medical domain, many categories)
  - Impressive results when only few training examples are available, e.g. using **88** training examples and testing a set of **3,299** examples, TSVM achieves the same accuracy that Naive Bayes (trained on the whole corpus of **9,603** examples!)

Transductive SVMs

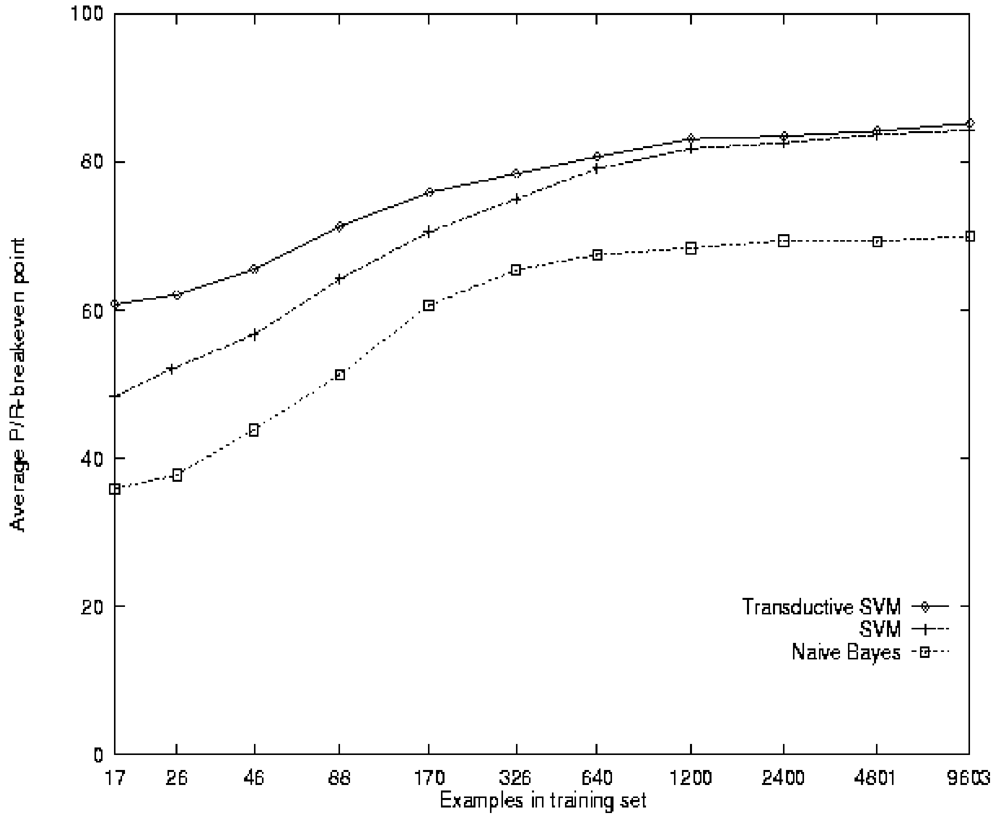


Figure 6: Average P/R-breakeven point on the Reuters dataset for different training set sizes and a test set size of 3,299.

(Joachims, 1999)

Transductive SVMs

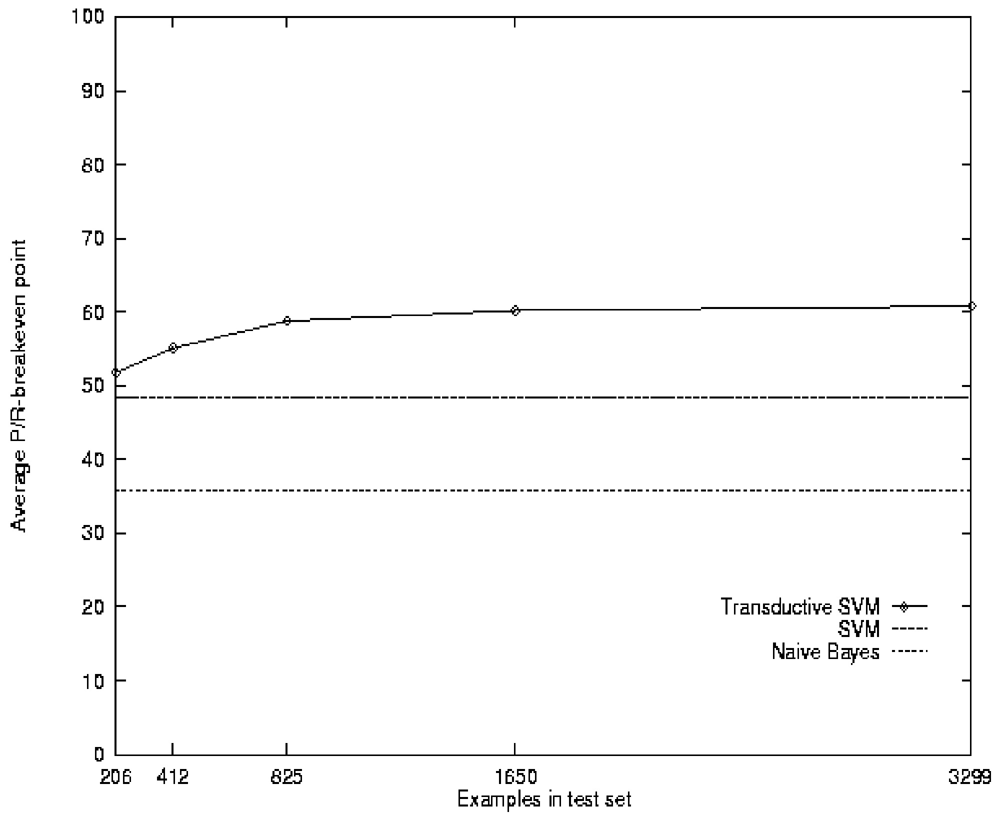


Figure 7: Average P/R-breakeven point on the Reuters dataset for 17 training documents and varying test set size for the TSVM.

(Joachims, 1999)

The End

Conclusions

- *General and rich class of pattern recognition methods*
- *Computationally efficient, statistically stable, and especially: versatile*
- *Very effective for a wide range of practical problems*
- *Much more than a replacement for Neural Networks*
- *Theoretical bounds for empirical error*