Support Vector Machines
(Chapter 7)

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

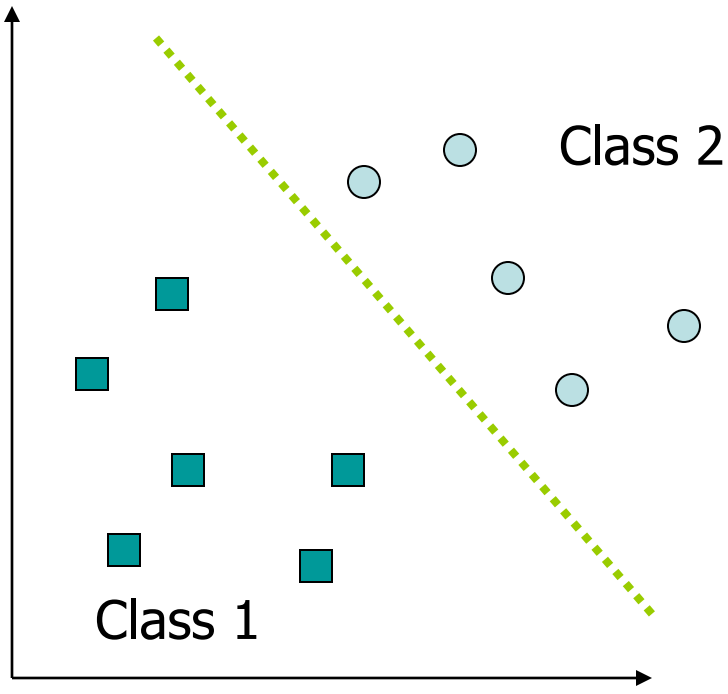
[1] B.E. Boser *et al*. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.

[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.

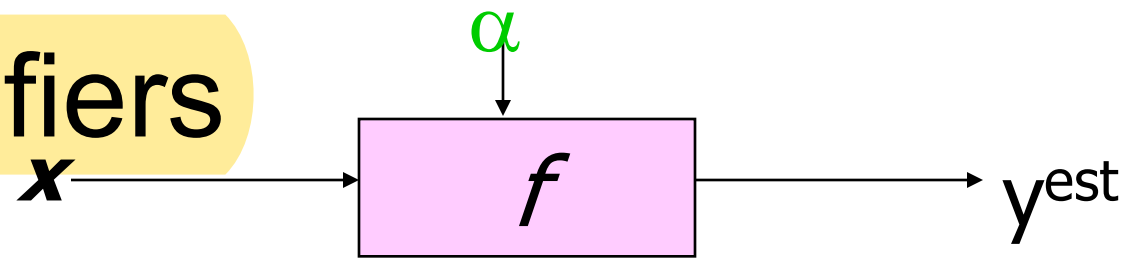
[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

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
- Are all decision boundaries equally good?

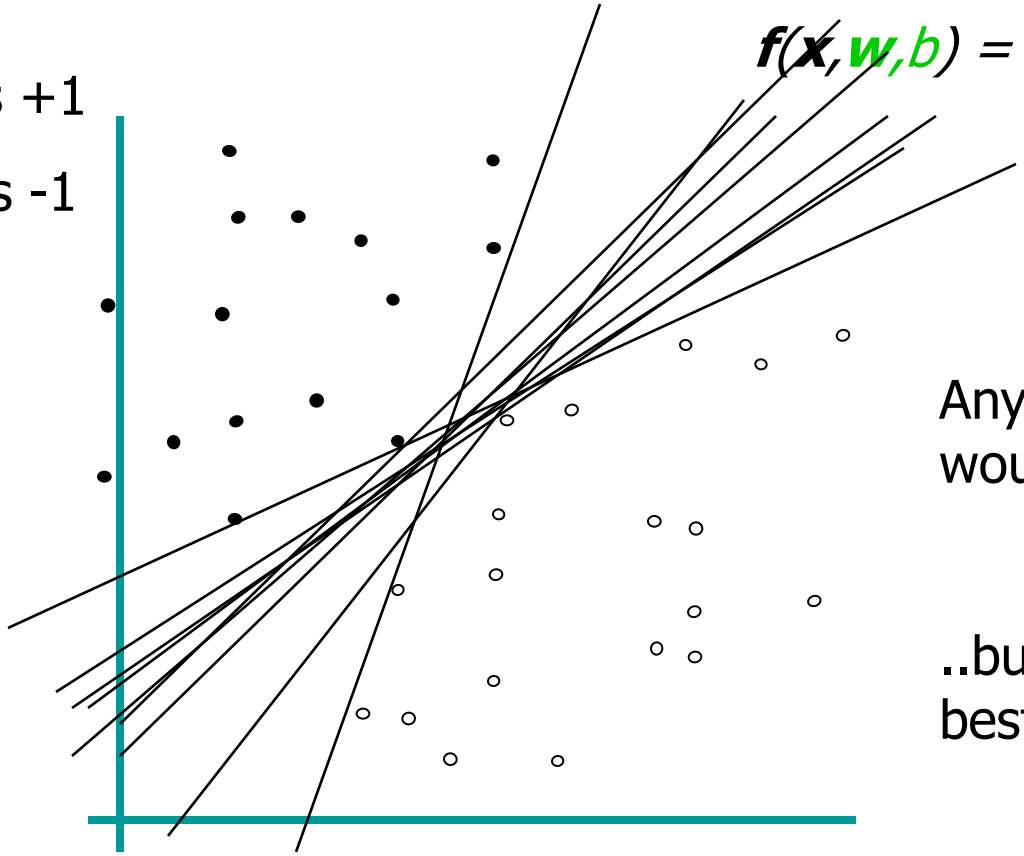


Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} + b)$

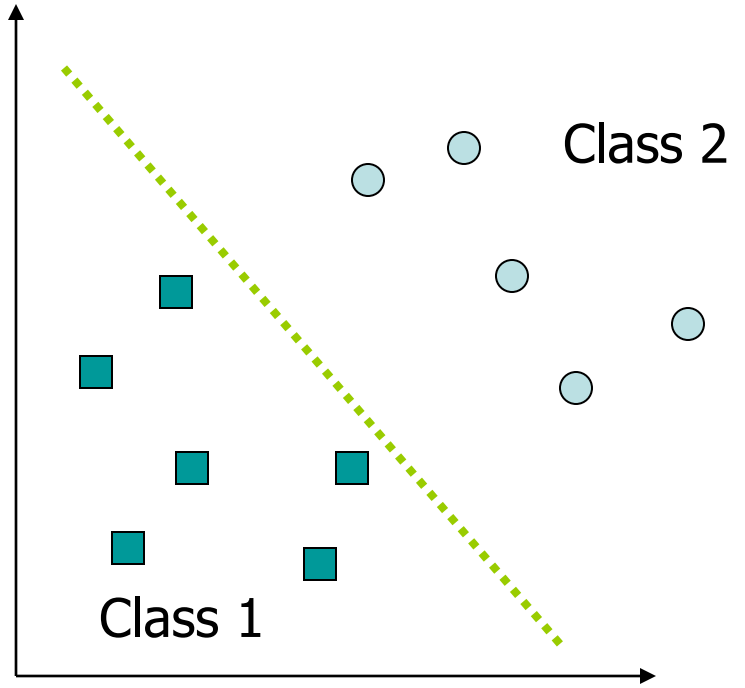
- denotes +1
- denotes -1

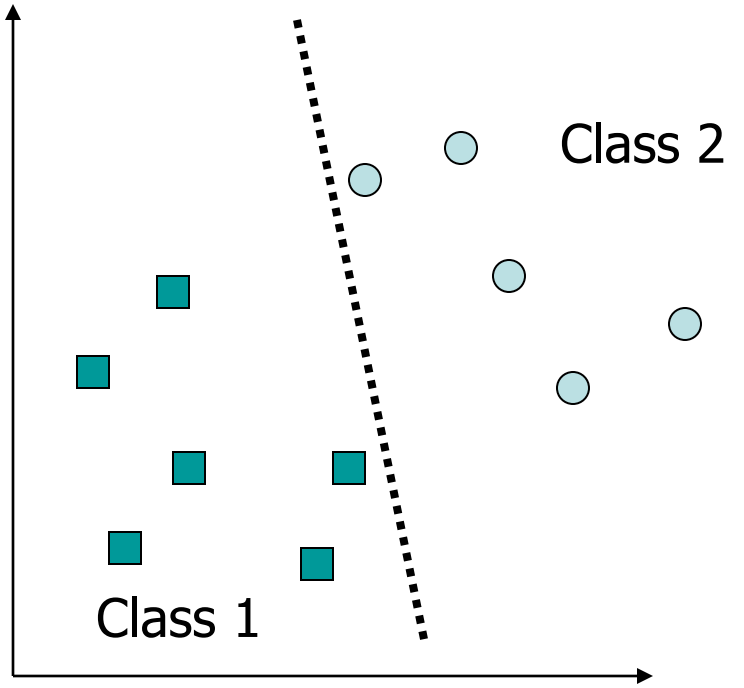


Any of these would be fine..

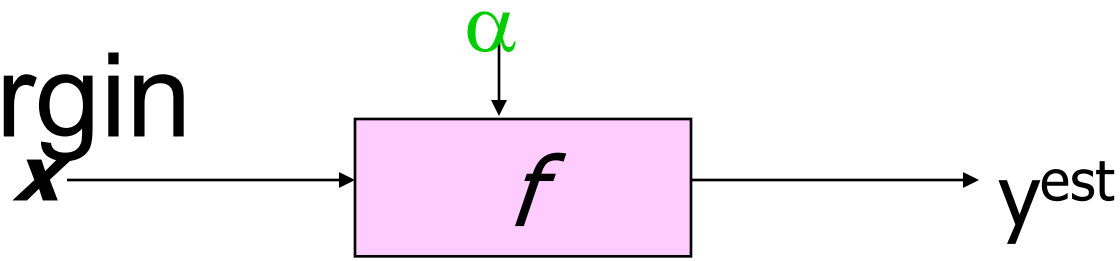
..but which is best?

Examples of Bad Decision Boundaries



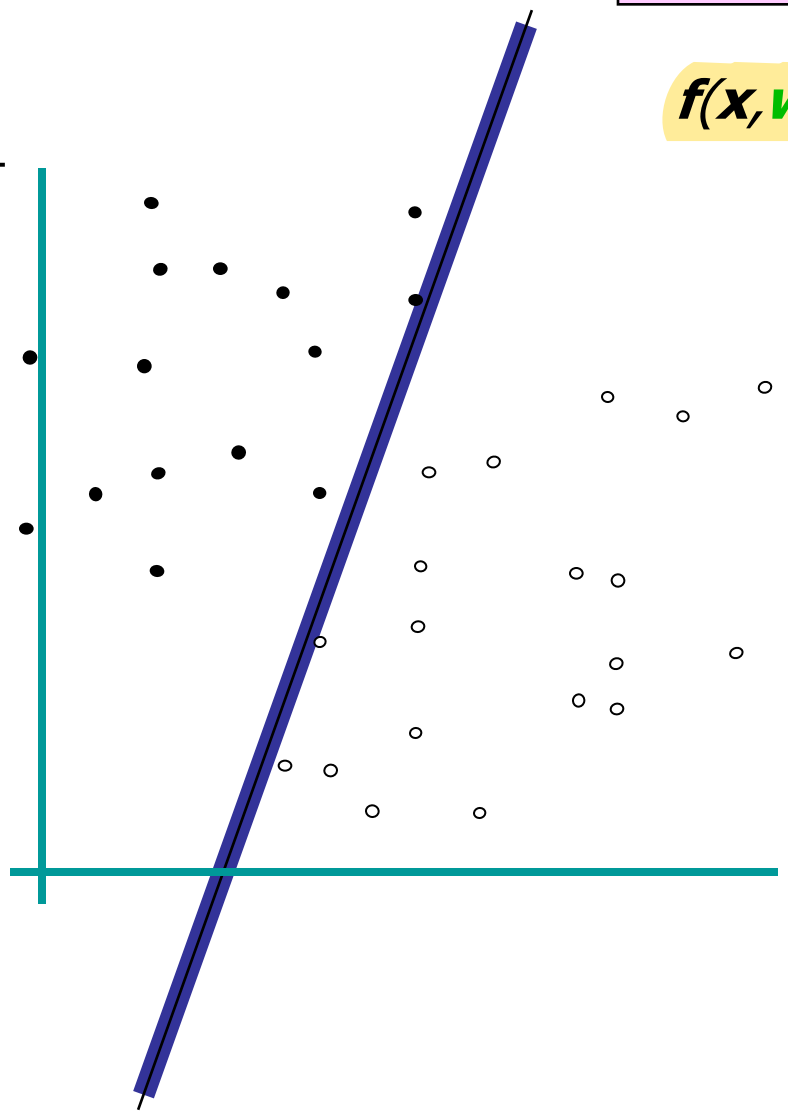


Classifier Margin



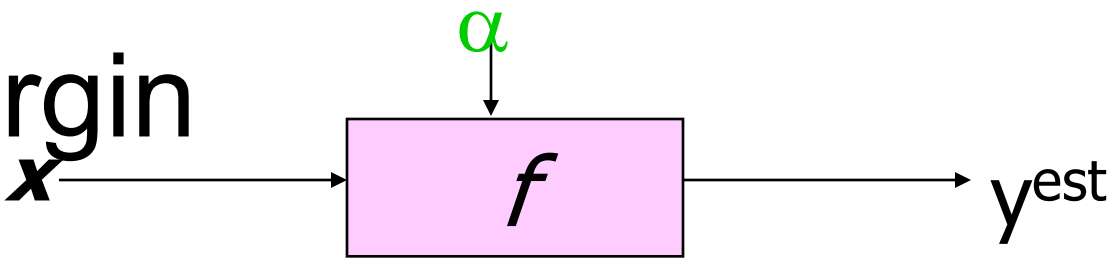
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} + b)$

- denotes +1
- denotes -1



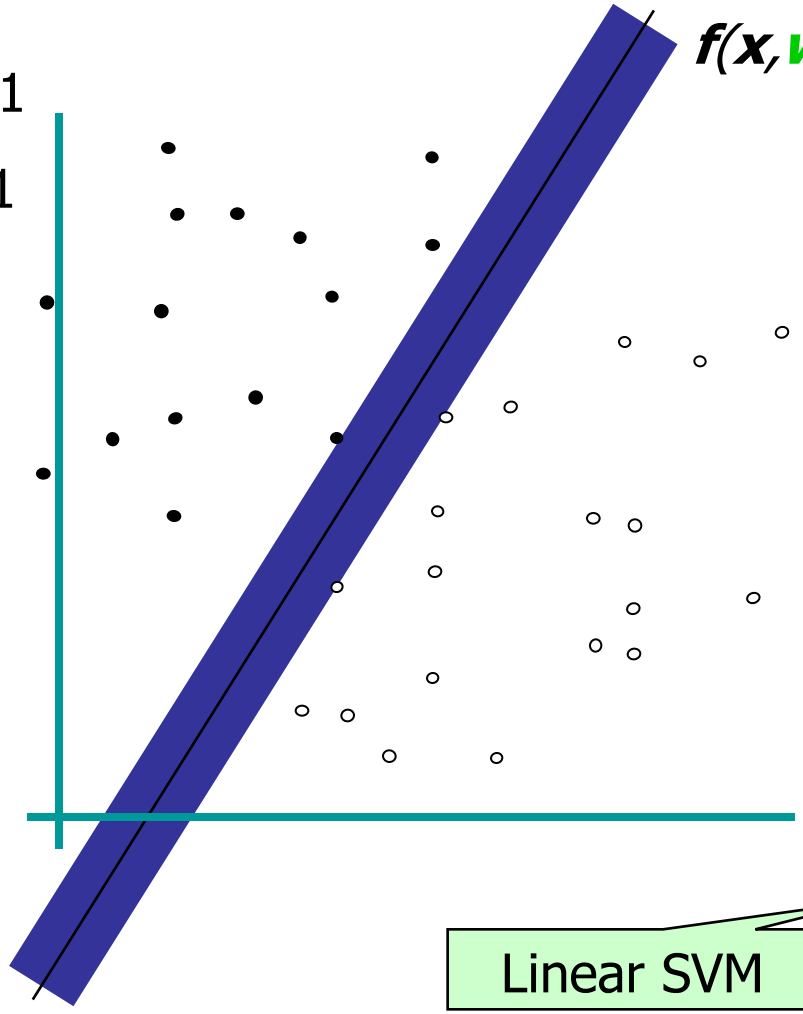
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} + b)$

- denotes +1
- denotes -1

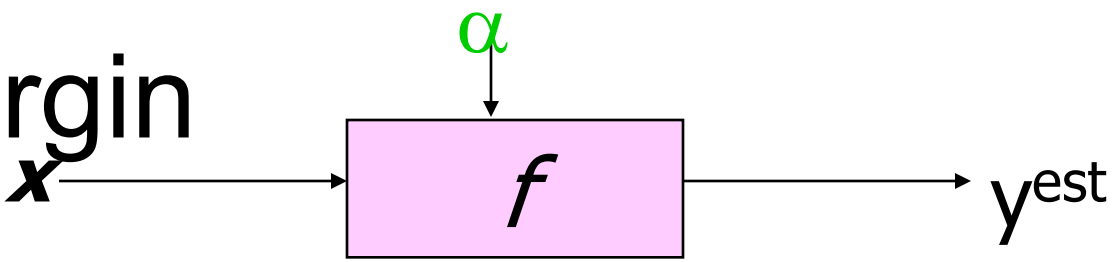


The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Maximum Margin



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} - b)$

- denotes +1
- denotes -1

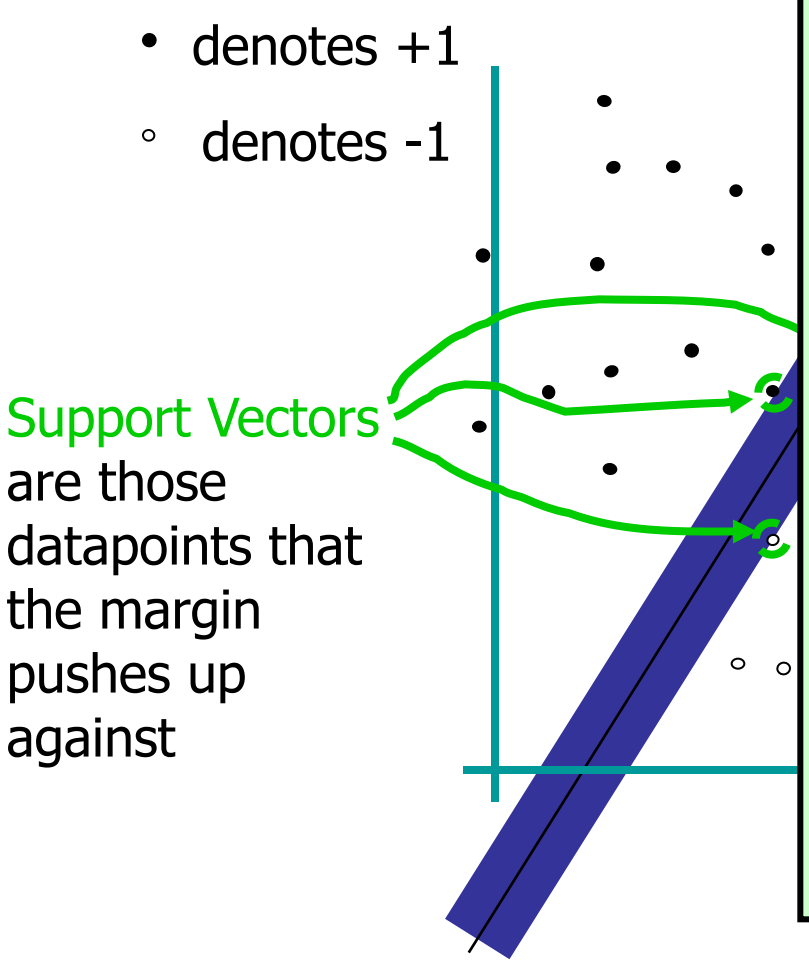
Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

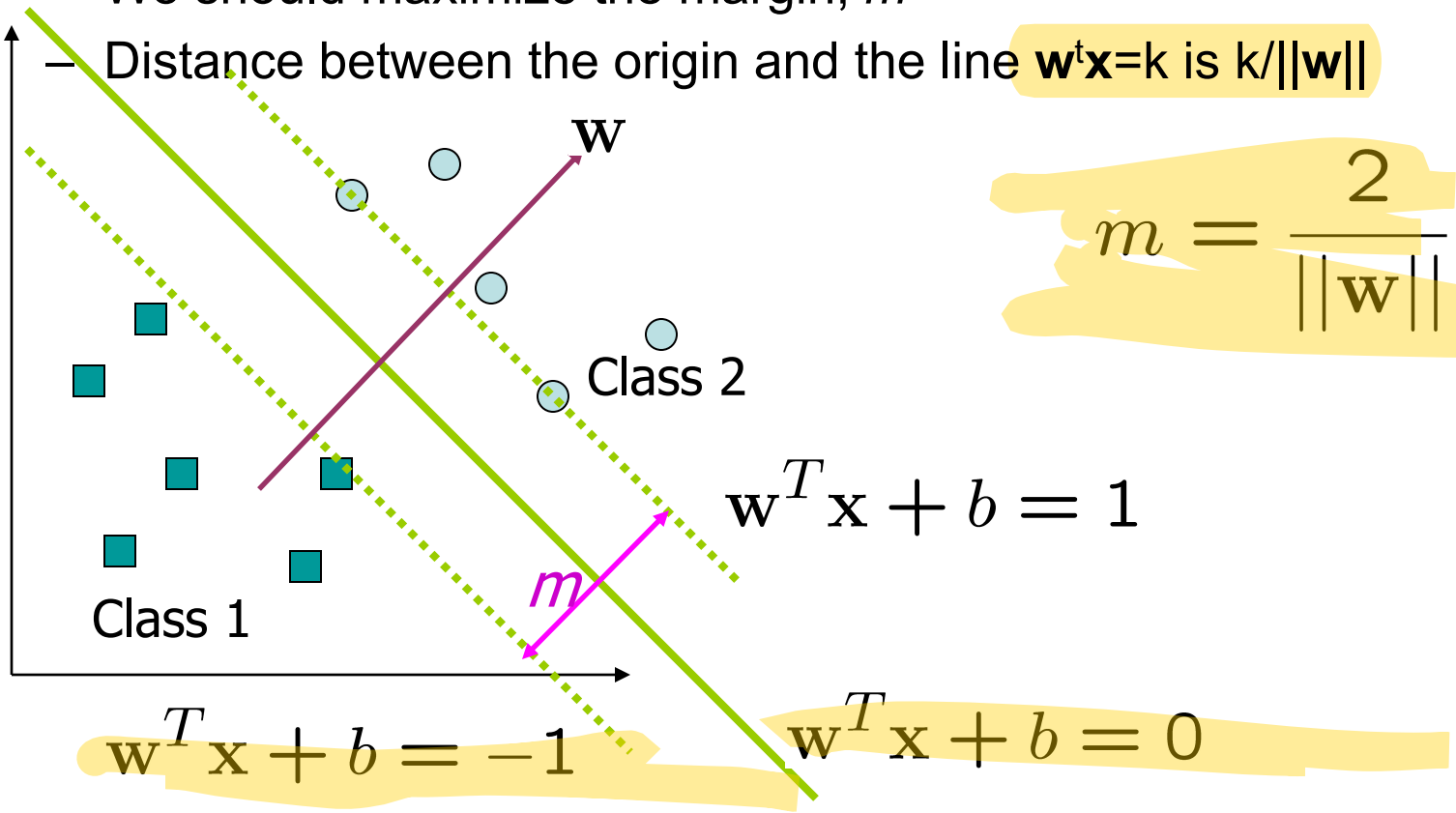
Why Maximum Margin?



1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
5. Empirically it works very very well.

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 $k/||\mathbf{w}||$



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

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 $\Rightarrow$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

- The decision boundary can be found 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

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

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}} \left(f(\mathbf{x}) + \sum_i \alpha_i g_i(\mathbf{x}) \right) \Big|_{\mathbf{x}=j x_0} = \mathbf{0} \\ g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \ldots, m \end{cases}$$

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

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

Ready for the kernel trick!

- 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 $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$

Karush-Kuhn-Tucker conditions

$Max \;\; f(x)$

subject to $g(x) \geq 0$

Optimize $L(x, \lambda) = f(x) + \lambda g(x)$

with respect to x, λ

subject to the conditions

$g(x) \geq 0$

$\lambda \geq 0$

$\lambda g(x) = 0$

Karush-Kuhn-Tucker conditions

For our problem

$$g(x_n) = y_n y(x_n) - 1 \geq 0 \qquad y(x_n) = \sum_{k=1}^{N} \alpha_n x_n^T x_k + b$$

$$\alpha_n \geq 0$$

$$\alpha_n (y_n y(x_n) - 1) = 0$$

either $\alpha_n = 0$ or $y_n y(x_n) = 1$

Any other point (not a support vector)

Points that push against the margin (support vectors) – the only ones with $\alpha_n \neq 0$

Consequently…

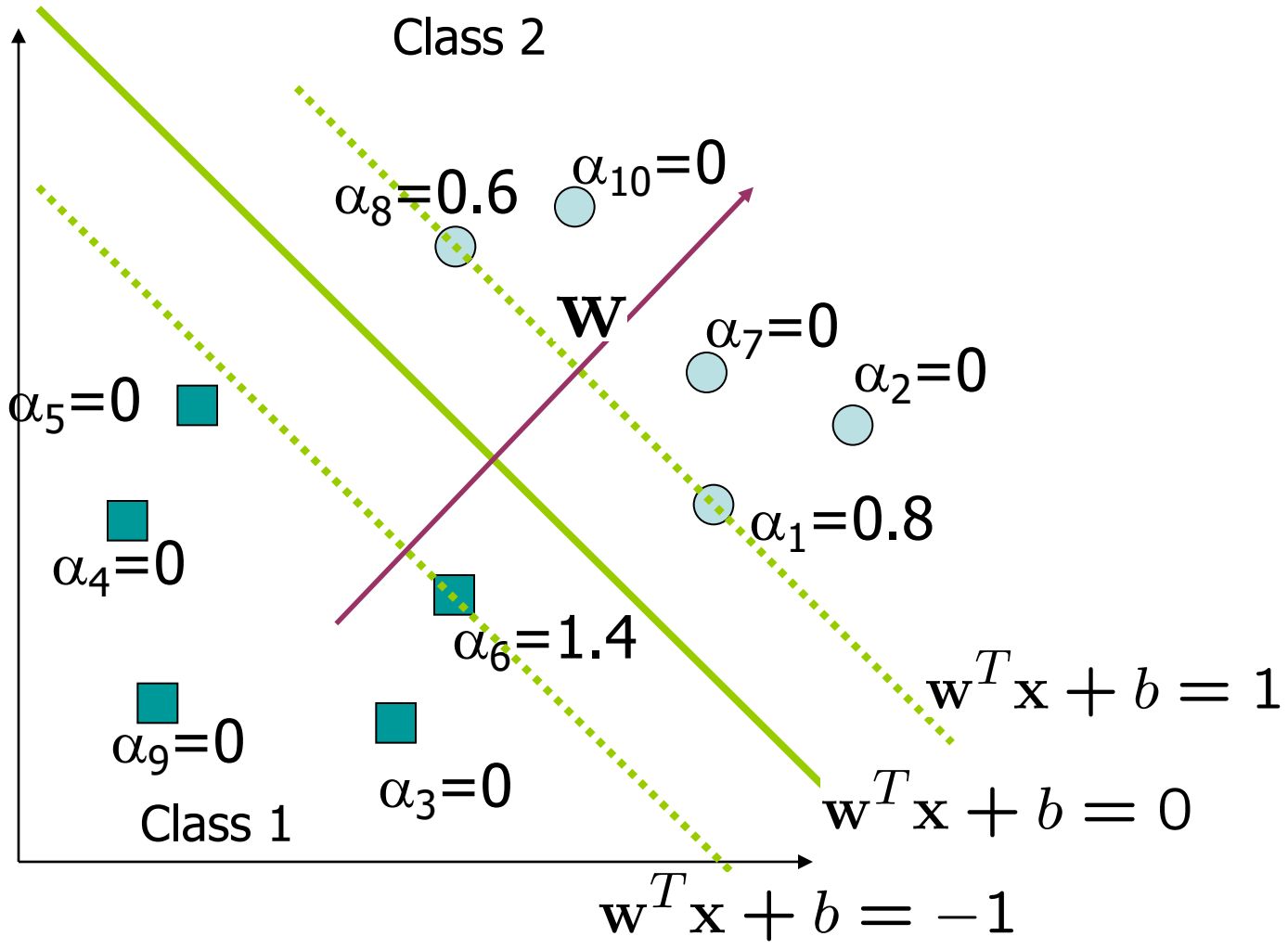
- The only points with positive Lagrange coefficients are support vectors
- The other points play no role (they don't add anything to the evaluation)
- You only need to retain support vectors, once the model is trained! (Huge savings!)

$$y(x) = \sum_{x_k \text{ a support vector}} \alpha_n x^T x_k + b$$

The Quadratic Programming Problem

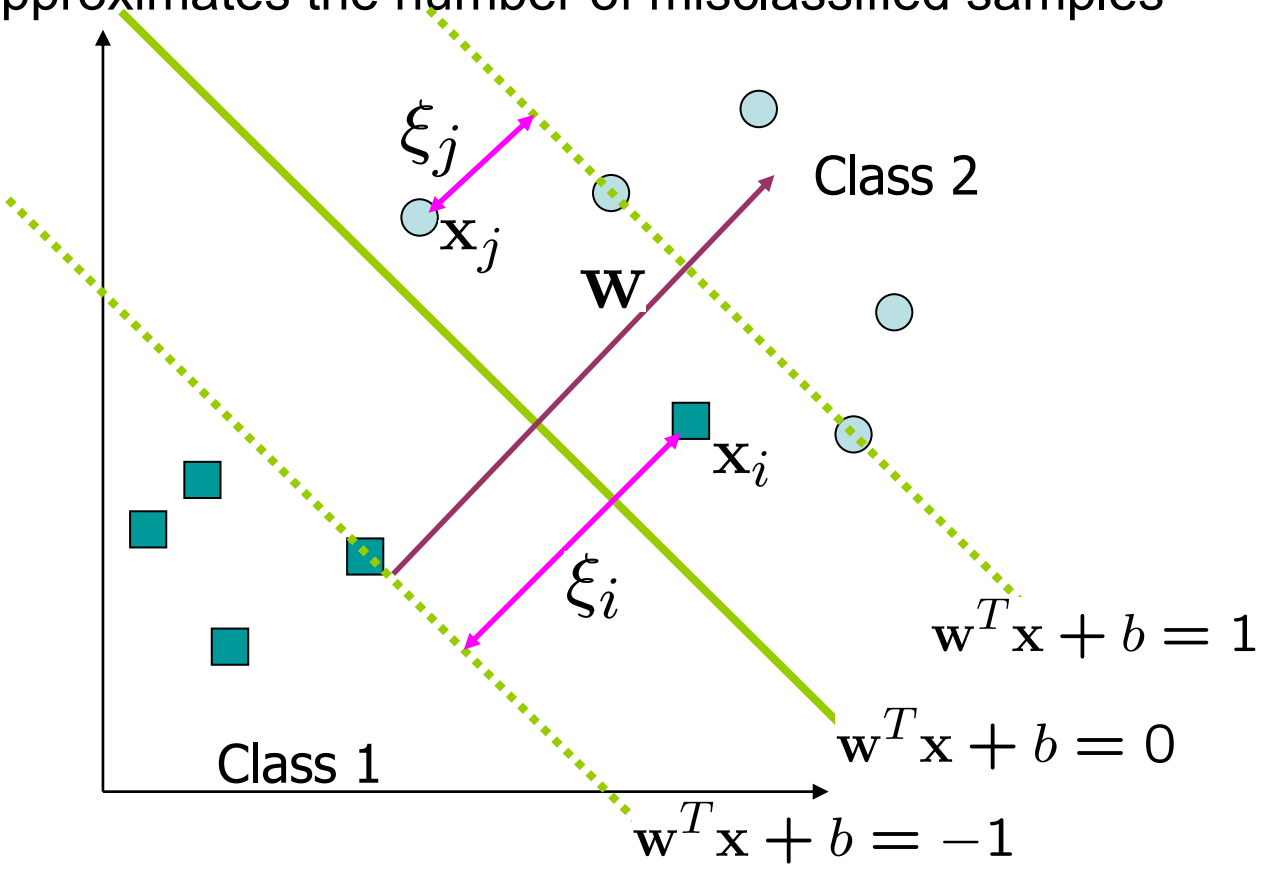
- Many approaches have been proposed
  - Loqo, cplex, etc. (see http://www.numerical.rl.ac.uk/qp/qp.html)
- Most are “interior-point” methods
  - Start with an initial solution that can violate the constraints
  - Improve this solution by optimizing the objective function and/or reducing the amount of 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

A Geometrical Interpretation



Overlapping: 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

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

Lagrangian (Primal problem)

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n\{y_n y(x_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n$$

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

KKT conditions

KKT

$$\alpha_n \geq 0$$

$$y_n y(x_n) - 1 + \xi_n \geq 0$$

$$\alpha_n (y_n y(x_n) - 1 + \xi_n) = 0$$

$$\mu_n \geq 0$$

$$\xi_n \geq 0$$

$$\mu_n \xi_n = 0$$

$$\text{also } \frac{\partial L}{\partial \xi_n} = 0 \Rightarrow \alpha_n = C - \mu_n$$

Only support vectors have positive alphas (discard the rest!)

If $\alpha_n < C \Rightarrow \mu_n > 0 \Rightarrow \xi_n = 0$

then the point is on the margin!

Else

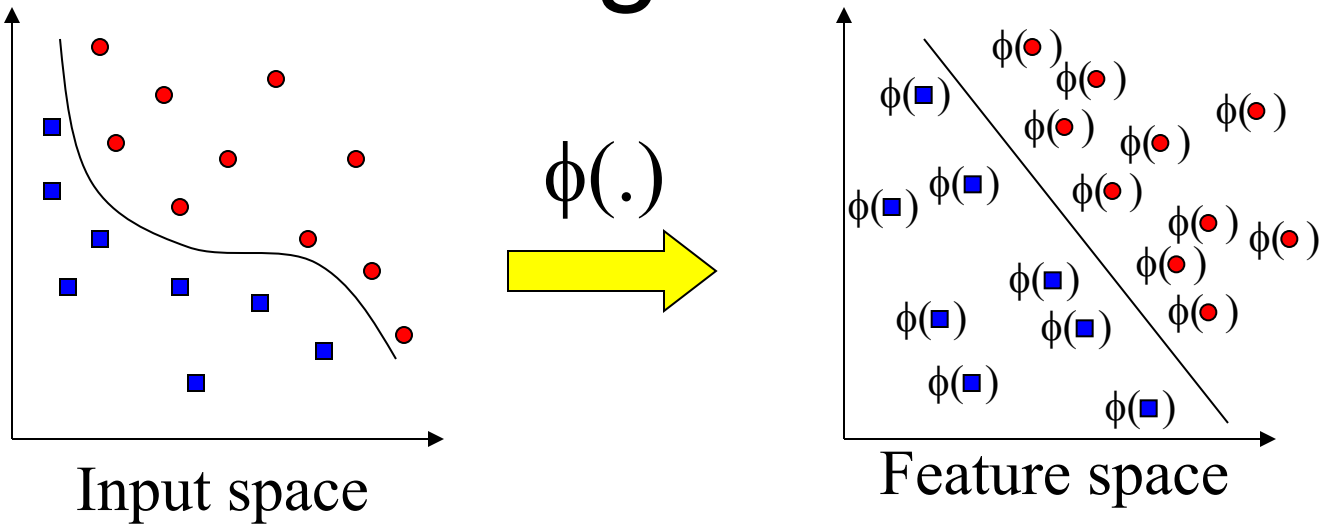
$\alpha_n = C \Rightarrow \mu_n = 0 \Rightarrow \xi_n \neq 0$

The point is inside the margin

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



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 can be infinite-dimensional!
- The kernel trick comes to rescue

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

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

An Example for $\phi(.)$ and K(.,.)

- Suppose $\phi(.)$ is given as follows

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

- An inner product in the feature space is

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

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

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

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

Kernel Functions

- In practical use of SVM, the user specifies the kernel function; the transformation $\phi(.)$ is not explicitly stated
- Given a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, the transformation $\phi(.)$ is given by its eigenfunctions (a concept in functional analysis)
  - Eigenfunctions can be difficult to construct explicitly
  - This is why people only specify the kernel function without worrying about the exact transformation
- Another view: kernel function, being an inner product, is really a similarity measure between the objects

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

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

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

With kernel function

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

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

Modification Due to Kernel Function

- For evaluation, 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

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

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

The parameter C for non-linear SVMs

- Perfect separation of training data can, in general be achieved in the enlarged feature space F
- Such perfect separation is actually bad: it leads to the danger of overfitting the data. Then the classifier will generalize poorly.
- A proper setting of C allows to avoid overfitting

C in non-linear SVMs

- Let's look once again at the objective function

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$$

- C is the penalty for errors of misclassification
  - Large C: discourages any positive parameters ξ_i $\Rightarrow$ tendency to overfit the data $\Rightarrow$highly complicated boundaries in input space
  - Small C: encourages a small value of ||w|| $\Rightarrow$ larger margin $\Rightarrow$more data on the wrong side of the margin $\Rightarrow$ smoother decision boundary in input space
  - In practice: tune C to achieve best performance

Alternative formulation of soft margin problems

- Due to Schölkopf et al

Primal problem:

$$\text{Min } \frac{1}{2}\|\mathbf{w}\|^2 - \upsilon\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i$$

Dual:

$$\text{Max } L(\mathbf{a}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m k(\mathbf{x}_n, \mathbf{x}_m)$$

$$s.t.$$

$$0 \le \alpha_n \le 1/N$$

$$\sum_{n=1}^{N}\alpha_n y_n = 0$$

$$\sum_{n=1}^{N}\alpha_n \ge \upsilon$$

called the $\upsilon - SVM$

Lower bound on the fraction of support vectors

Upper bound on the fraction of margin errors

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 $\mathbf{z}$, 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 measures 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

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

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

Example

- By using a QP solver, we get
  - α_1=0, α_2=2.5, α_3=0, α_4=7.333, α_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

α_5 y_5 $K(z, x_5)$

$$f(z) = 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + 4.833(1)(6z+1)^2 + b$$
$$= 0.6667z^2 - 5.333z + b$$

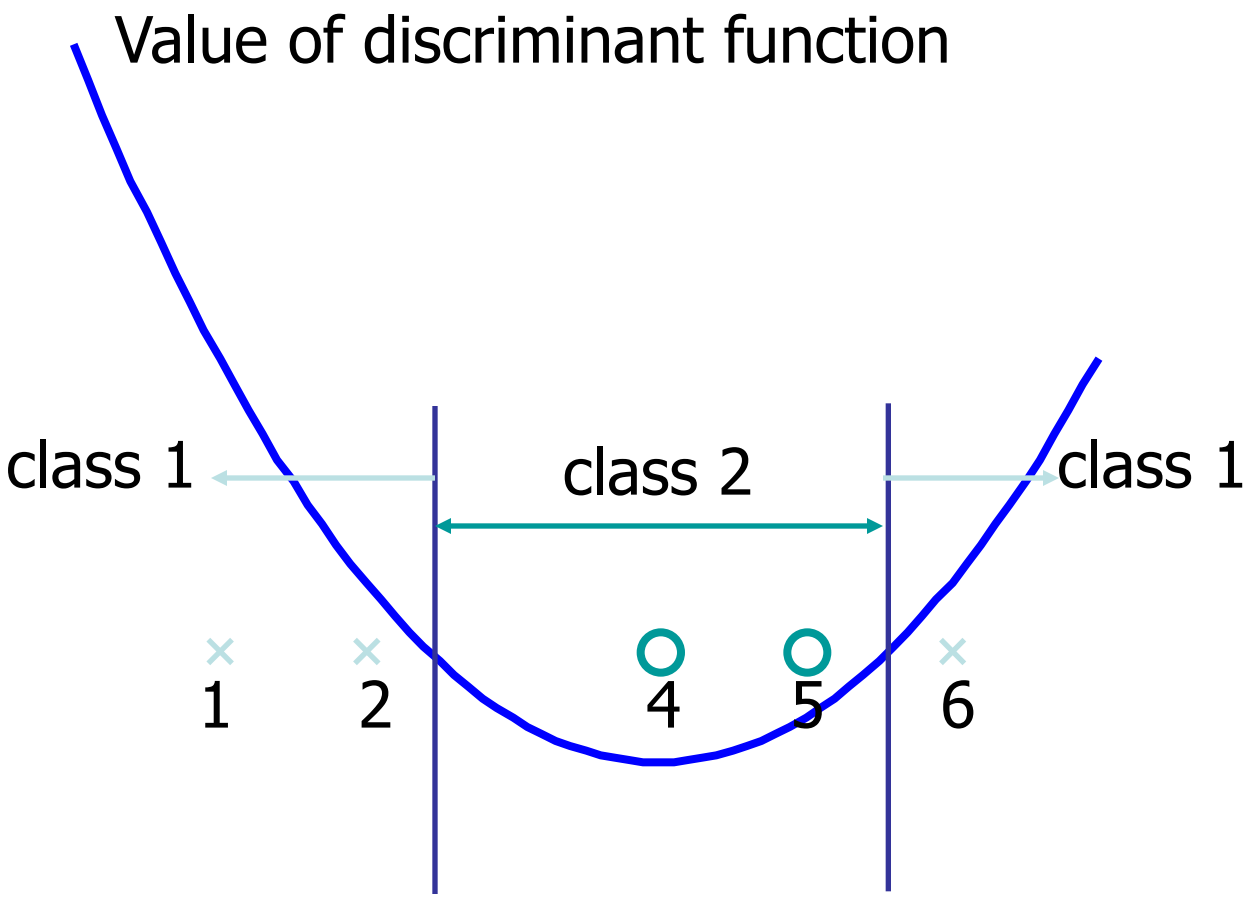
- *b* is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2 and x_5 lie on the line $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = 1$

and x_4 lies on the line $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = -1$

- All three give b=9

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

Example

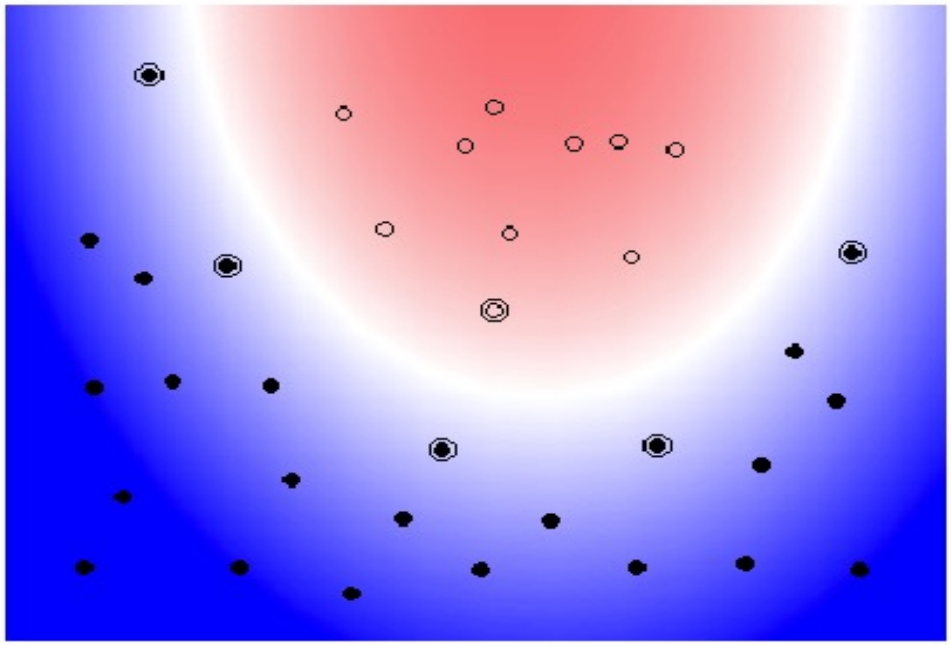

Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6

Nonlinear Kernel (I)

Example: SVM with Polynomial of Degree 2

Kernel: $K(\vec{x}_i, \vec{x}_j) = [\vec{x}_i \cdot \vec{x}_j + 1]^2$

plot by Bell SVM applet

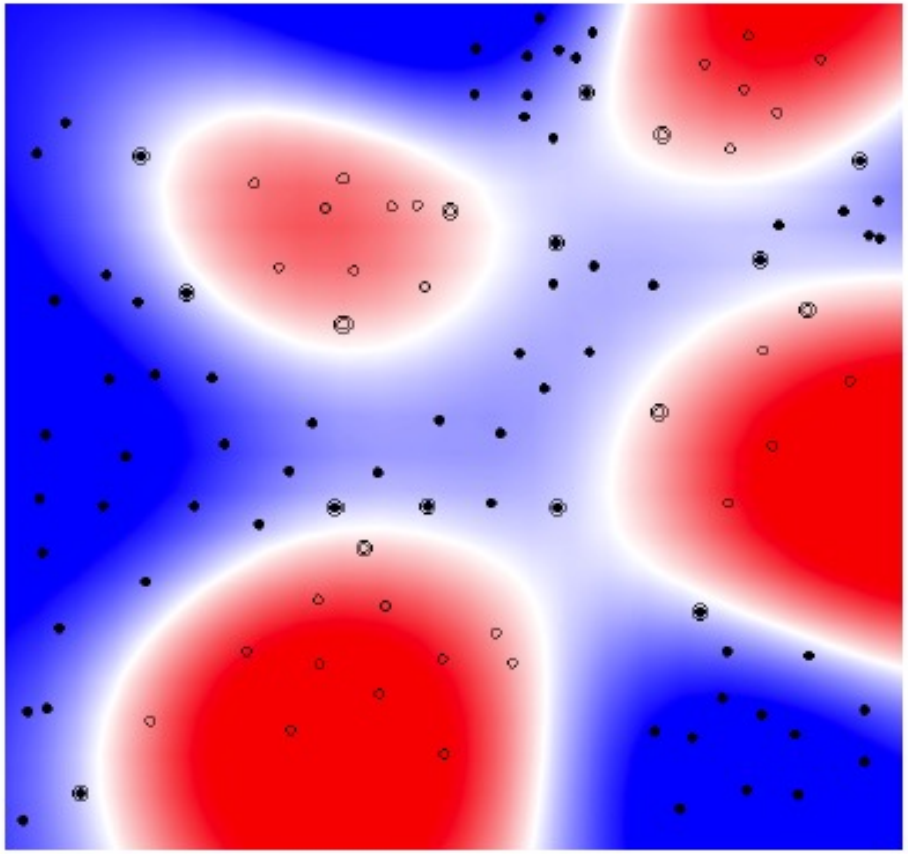


Nonlinear Kernel (II)

Example: SVM with RBF-Kernel

Kernel: $K(\vec{x}_i, \vec{x}_j) = \exp(-|\vec{x}_i - \vec{x}_j|^2 / \sigma^2)$

plot by Bell SVM applet

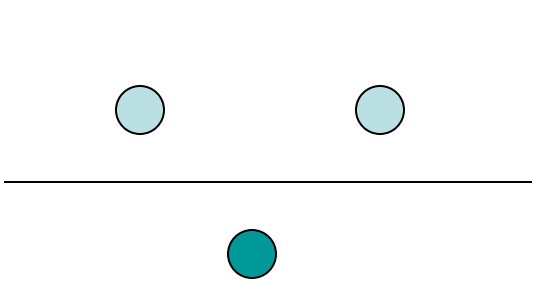


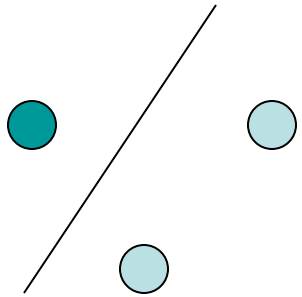
Why SVM Work?

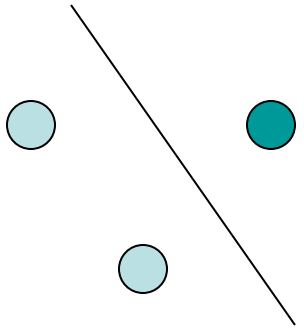
- The feature space is often very high dimensional. Why don't we have the curse of dimensionality?
- A classifier in a high-dimensional space has many parameters and is hard to estimate
- Vapnik argues that the fundamental problem is not the number of parameters to be estimated. Rather, the problem is about the flexibility of a classifier
- Typically, a classifier with many parameters is very flexible, but there are also exceptions
  - Let $x_i=10^i$ where i ranges from 1 to n. The classifier $y = \text{sign}\left(\sin(\alpha x)\right)$ can classify all x_i correctly for all possible combination of class labels on x_i
  - This 1-parameter classifier is very flexible

Why SVM works?

- Vapnik argues that the flexibility of a classifier should not be characterized by the number of parameters, but by the flexibility (capacity) of a classifier
  - This is formalized by the “VC-dimension” of a classifier
- Consider a linear classifier in two-dimensional space
- If we have three training data points, no matter how those points are labeled, we can classify them perfectly

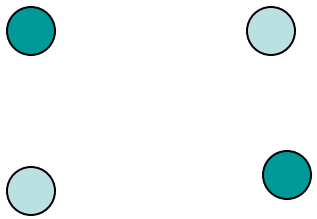






VC-dimension

- However, if we have four points, we can find a labeling such that the linear classifier fails to be perfect



- We can see that 3 is the critical number
- The VC-dimension of a linear classifier in a 2D space is 3 because, if we have 3 points in the training set, perfect classification is always possible irrespective of the labeling, whereas for 4 points, perfect classification can be impossible

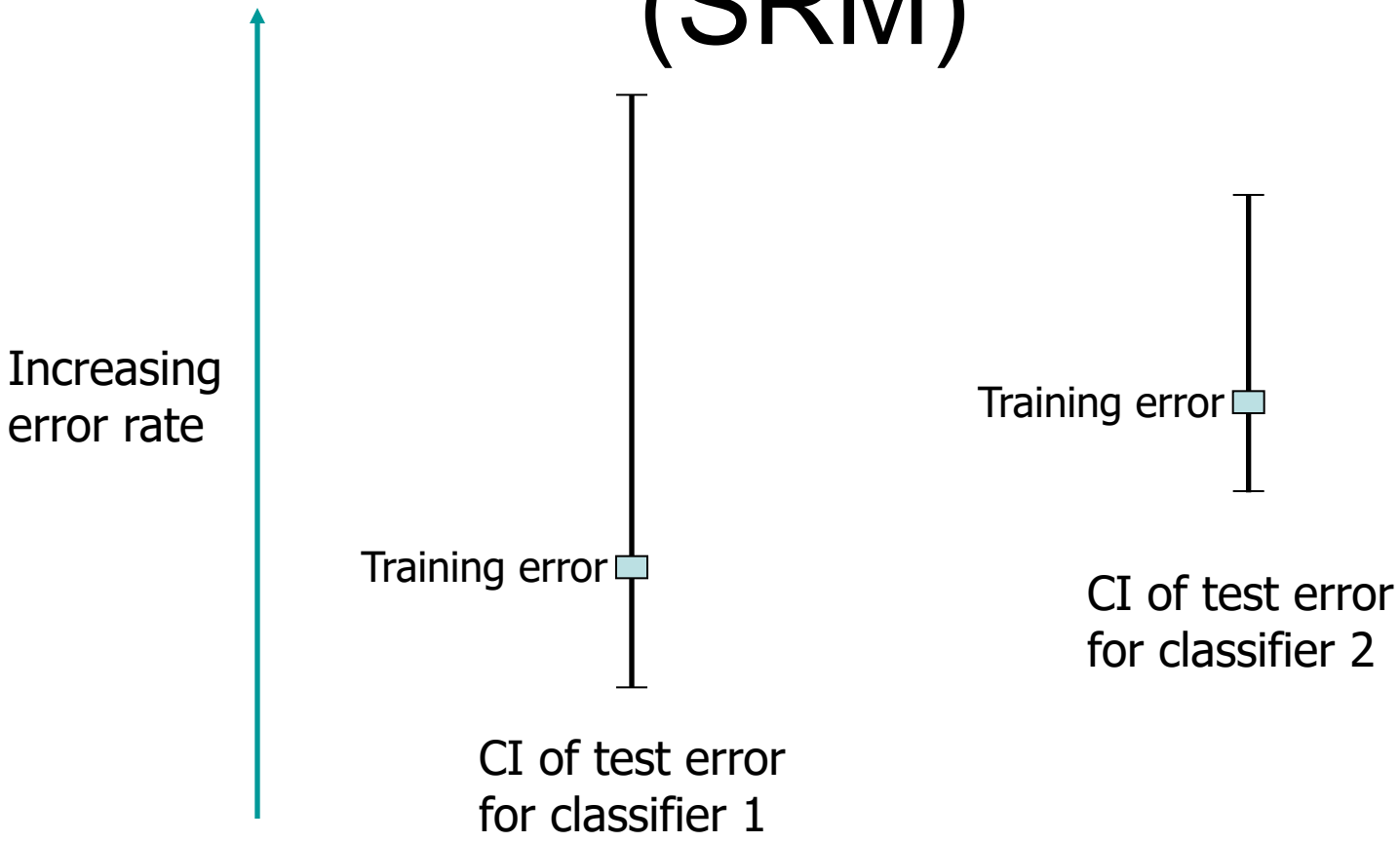
VC-dimension

- The VC-dimension of the nearest neighbor classifier is infinity, because no matter how many points you have, you get perfect classification on training data
- The higher the VC-dimension, the more flexible a classifier is
- VC-dimension, however, is a theoretical concept; the VC-dimension of most classifiers, in practice, is difficult to be computed exactly
  - Qualitatively, if we think a classifier is flexible, it probably has a high VC-dimension

Structural Risk Minimization (SRM)

- A fancy term, but it simply means: we should find a classifier that minimizes the sum of training error (empirical risk) and a term that is a function of the flexibility of the classifier (model complexity)
- Recall the concept of confidence interval (CI)
  - For example, we are 99% confident that the population mean lies in the 99% CI estimated from a sample
- We can also construct a CI for the generalization error (error on the test set)

Structural Risk Minimization (SRM)



- SRM prefers classifier 2 although it has a higher training error, because the upper limit of CI is smaller

Structural Risk Minimization (SRM)

- It can be proved that the more flexible a classifier, the "wider" the CI is
- The width can be upper-bounded by a function of the VC-dimension of the classifier
- In practice, the confidence interval of the testing error contains [0,1] and hence is trivial
  - Empirically, minimizing the upper bound is still useful
- The two classifiers are often "nested", i.e., one classifier is a special case of the other
- SVM can be viewed as implementing SRM because $\sum_i \xi_i$ approximates the training error; $\frac{1}{2}||w||^2$ is related to the VC-dimension of the resulting classifier
- See http://www.svms.org/srm/ for more details

Justification of SVM

- Large margin classifier
- SRM
- Ridge regression: the term $\frac{1}{2}||w||^2$ "shrinks" the parameters towards zero to avoid overfitting
- The term the term $\frac{1}{2}||w||^2$ can also be viewed as imposing a weight-decay prior on the weight vector, and we find the MAP estimate

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.

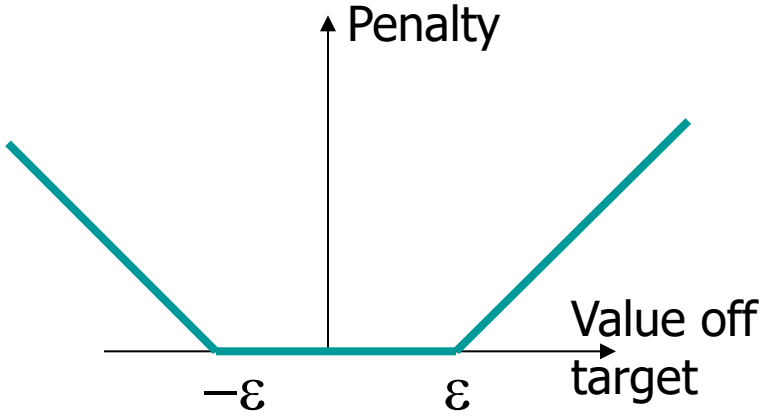
Other Types of Kernel Methods

- 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

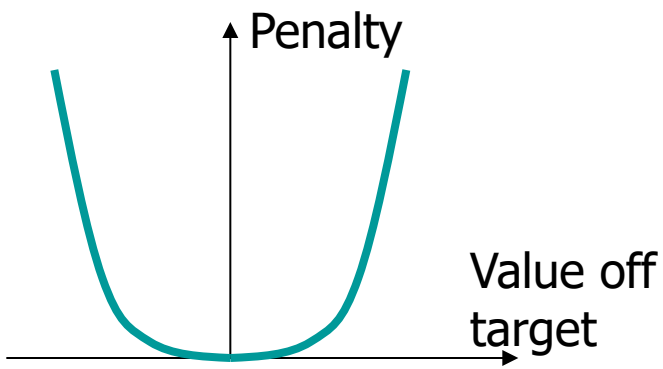
Epsilon Support Vector Regression (ε-SVR)

- Linear regression in feature space
- Unlike in least square regression, the error function is ε-insensitive loss function
  - Intuitively, mistake less than ε is ignored
  - This leads to sparsity similar to SVM

ε-insensitive loss function



Square loss function



Epsilon Support Vector Regression (ε-SVR)

- Given: a data set $\{x_1, ..., x_n\}$ with target values $\{u_1, ..., u_n\}$, we want to do ε-SVR
- The optimization problem is

$$\text{Min } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\text{subject to } \begin{cases} u_i - \mathbf{w}^T\mathbf{x}_i - b \leq \epsilon + \xi_i \\ \mathbf{w}^T\mathbf{x}_i + b - u_i \leq \epsilon + \xi_i^* \\ \xi_i \geq 0, \xi_i^* \geq 0 \end{cases}$$

- Similar to SVM, this can be solved as a quadratic programming problem

Epsilon Support Vector Regression (ε-SVR)

- *C* is a parameter to control the amount of influence of the error
- The ½||w||^2 term serves as controlling the complexity of the regression function
  - This is similar to ridge regression
- After training (solving the QP), we get values of α_i and α_i^*, which are both zero if $\mathbf{x}_i$ does not contribute to the error function
- For a new data **z**,

$$f(\mathbf{z}) = \sum_{j=1}^{s} (\alpha_{t_j} - \alpha_{t_j}^*) K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

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