Machine Learning

Lecture: Support Vector Machines

Hiroshi Shimodaira

16 February 2024

Ver. 1.0

Questions you should be able to answer after this week

- What is Support Vector Machine (SVM)?
- Training (optimisation problem) of linear SVM?
  What is maximum margin
- How to solve the optimisation problem?
- What are the support vectors?
- What is soft-margin SVM (SVM with slack variables)?
- How to make non-linear SVM?
- What is kernel and what is kernel trick?
- What are pros and cons with SVM?
- What applications are SVM successful for?

History of machine learning

18c	Naive Bayes classifier
1940s	Threshold logic - Warren McCulloch and Walter Pitts
	Logistic regression - Joseph Berkson
1951	k-NN - Evelyn Fix and Joseph Hodges
1957	Perceptron - Frank Rosenblatt
1959	Decision tree - William Belson (?)
1986	ANN with EBP - D.Rumelhart, G.Hinton, and R.Williams



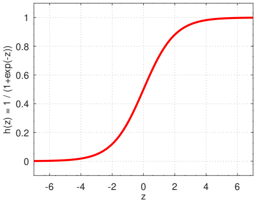
1993-97	Support Vector Machine - Vladimir Vapnik

after Yann Lu Cun's photo galleries

Recap – Logistic Regression

- $P(Y=1|\boldsymbol{x}) = \dfrac{1}{1+\exp\left(-(\boldsymbol{w}^T\boldsymbol{x}+w_0)\right)}$

  $\boldsymbol{x} = (x_1, , \ldots, x_d)^T,$
  $\boldsymbol{w} = (w_1, \ldots, w_d)^T, Y \in \{0, 1\}$

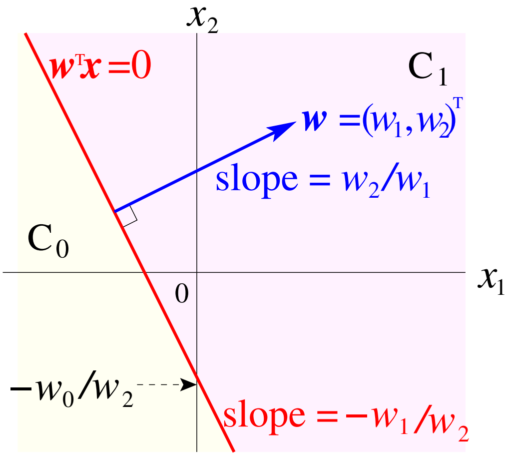


- Training on a data set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ based on maximum likelihood estimation (MLE):

$$\max_{\boldsymbol{w}, w_0} \prod_{i=1}^{n} P(Y=y_i|\boldsymbol{x}_i)$$

Decision boundary and decision regions

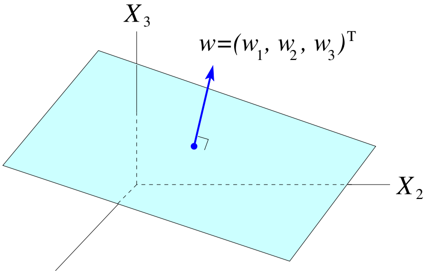
$$P(Y=1|\boldsymbol{x}) = \frac{1}{1+\exp\left(-(\boldsymbol{w}^T\boldsymbol{x}+w_0)\right)} \quad \rightarrow \quad \text{decision boundary: } \boldsymbol{w}^T\boldsymbol{x}+w_0=0$$



Decision boundary and decision regions (*cont.*)

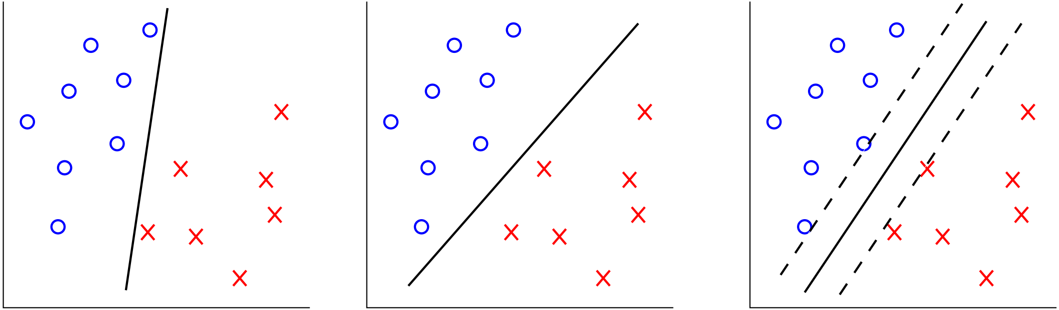
$$P(Y=1|\boldsymbol{x}) = \frac{1}{1+\exp\left(-(\boldsymbol{w}^T\boldsymbol{x}+w_0)\right)}$$

Dimension	Decision boundary	
2	line	$w_1x_1 + w_2x_2 + w_0 = 0$
3	plane	$w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0$
$\vdots$		
d	hyperplane	$\left(\sum_{i=1}^{d} w_ix_i\right) + w_0 = 0$



Linear classifiers and large margin classifiers

$\hat{y}(\boldsymbol{x}) = f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$



(a) (b) (c)

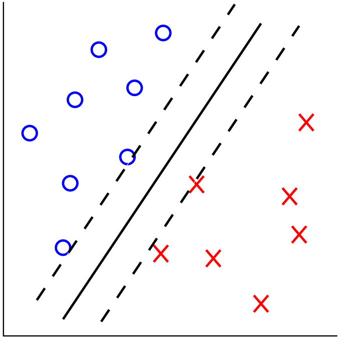
$\boldsymbol{w}^T\boldsymbol{x}_i + w_0 = 0$

$\boldsymbol{w}^T\boldsymbol{x}_i + w_0 = +1$

$\boldsymbol{w}^T\boldsymbol{x}_i + w_0 = -1$

Large margin classifiers (*cont.*)

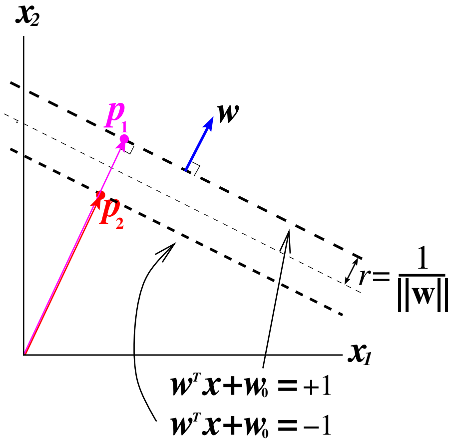
Proposed by several people, e.g. Vladimir Vapnik (1963, 1992)



(c)

$$\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \geq +1 \quad \forall i \text{ s.t. } y_i = +1$$
$$\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \leq -1 \quad \forall i \text{ s.t. } y_i = -1$$

Margin



$$
\begin{aligned}
\|\mathbf{p}_1 - \mathbf{p}_2\| &= \big|\|\mathbf{p}_1\| - \|\mathbf{p}_2\|\big| \\
&= \left| \frac{-w_0 + 1}{\|\mathbf{w}\|} - \frac{-w_0 - 1}{\|\mathbf{w}\|} \right| \\
&= \frac{2}{\|\mathbf{w}\|} = 2r
\end{aligned}
$$

where

$$
\begin{aligned}
1 &= \mathbf{w}^T \mathbf{p}_1 + w_0 \\
&= \|\mathbf{w}\|\|\mathbf{p}_1\| \cos(\theta)|_{\theta=0} + w_0 \\
&= \|\mathbf{w}\|\|\mathbf{p}_1\| + w_0 \\
\|\mathbf{p}_1\| &= \frac{-w_0 + 1}{\|\mathbf{w}\|}
\end{aligned}
$$

Support Vector Machine (SVM)

Training $\max_{\boldsymbol{w}} \quad \frac{1}{\|\boldsymbol{w}\|}$

$$\text{s.t.} \quad \boldsymbol{w}^T\boldsymbol{x}_i + w_0 \geq +1 \ \text{ for all } i \text{ with } y_i = +1$$
$$\boldsymbol{w}^T\boldsymbol{x}_i + w_0 \leq -1 \ \text{ for all } i \text{ with } y_i = -1$$

Equivalent to

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t.} \quad y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + w_0\right) \geq 1 \ \text{ for all } i$$

NB: $\boldsymbol{w}^T\boldsymbol{w} = \|\boldsymbol{w}\|^2$

NB: *constrained, quadratic and convex optimisation problem* $\rightarrow$ *no local mimima!*

Solution: $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i, \quad \alpha_i \geq 0 \ \cdots$ most of α_i are zeros normally

Those $\{\boldsymbol{x}_i\}$ whose $\alpha_i > 0$ are called *support vectors*.

Classification

$$g(\boldsymbol{x}) = \text{sgn}(\boldsymbol{w}^T\boldsymbol{x} + w_0) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i\, y_i\, \boldsymbol{x}_i^T\boldsymbol{x} + w_0\right)$$

Why $+1$ instead of $+\varepsilon$?

Assuming $\varepsilon > 0$,

$$\begin{aligned} \min_{\boldsymbol{w}, w_0} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_i\left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0\right) \geq \varepsilon \ \text{ for all } i \end{aligned}$$

$\Rightarrow$

$$\begin{aligned} \min_{\boldsymbol{w}, w_o} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_i\left(\frac{\boldsymbol{w}^T}{\varepsilon} \boldsymbol{x}_i + \frac{w_0}{\varepsilon}\right) \geq 1 \ \text{ for all } i \end{aligned}$$

Letting $\dot{w} = \frac{\mathbf{w}}{\varepsilon}$ and $\dot{w}_0 = \frac{w_0}{\varepsilon}$,

$$\begin{aligned} \min_{\boldsymbol{w}, w_0} \quad & \frac{\varepsilon^2}{2}\|\dot{\boldsymbol{w}}\|^2 \\ \text{s.t.} \quad & y_i\left(\dot{\boldsymbol{w}}^T \boldsymbol{x}_i + \dot{w}_0\right) \geq 1 \ \text{ for all } i \end{aligned}$$

Optimisation problems in SVM

$$\begin{aligned} \min_{\boldsymbol{w}, w_0} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ \text{s.t.} \quad & y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + w_0\right) \geq 1 \;\text{ for all } i \end{aligned}$$

Using the Lagrange multipliers $\alpha_i \geq 0$, the Lagrangian is given as:

$$L(\boldsymbol{\alpha}, \dot{\mathbf{w}}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1\right)$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and $\dot{\mathbf{w}} = (\mathbf{w}, w_0)$. The dual problem is defined as

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & L(\boldsymbol{\alpha}, \dot{\mathbf{w}}) \\ \text{s.t.} \quad & \boldsymbol{\alpha} \geq \mathbf{0} \end{aligned}$$

Optimisation problems in SVM *(cont.)*

$$L(\boldsymbol{\alpha}, \dot{\mathbf{w}}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1 \right)$$

$$\frac{\partial L(\boldsymbol{\alpha}, \dot{\mathbf{w}})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i\, y_i\, \boldsymbol{x}_i = \mathbf{0},$$

$$\frac{\partial L(\boldsymbol{\alpha}, \dot{\mathbf{w}})}{\partial w_0} = -\sum_{i=1}^{n} \alpha_i\, y_i = 0.$$

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

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

Optimisation problems in SVM *(cont.)*

Putting the results to the Lagrangian yields:

$$
\begin{aligned}
L(\boldsymbol{\alpha}, \dot{\mathbf{w}}) &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1 \right) \\
&= \frac{1}{2}\sum_{i,j=1}^{n} y_i\, y_j\, \alpha_i\, \alpha_j\, \mathbf{x}_i^T\mathbf{x}_j - \sum_{i,j=1}^{n} y_i\, y_j\, \alpha_i\, \alpha_j\, \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i \\
&= -\frac{1}{2}\sum_{i,j=1}^{n} y_i\, y_j\, \alpha_i\, \alpha_j\, \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i
\end{aligned}
$$

The necessary and sufficient conditions for $\mathbf{w}^*$ to be an optimum are:

$$
\frac{\partial L(\boldsymbol{\alpha}^*, \dot{\mathbf{w}}^*)}{\partial \mathbf{w}} = \mathbf{0}, \quad \frac{\partial L(\boldsymbol{\alpha}^*, \dot{\mathbf{w}}^*)}{\partial w_0} = 0, \quad \alpha_i^* \geq 0, \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1 \geq 0,
$$

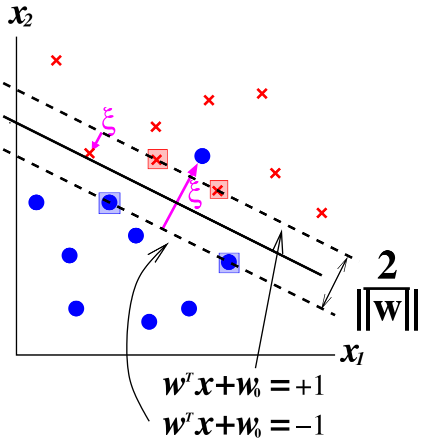
$\alpha_i^* \left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1 \right) = 0$, for all i $\cdots$ Karush-Kuhn-Tuckert (KKT) condition

which means that either $\alpha_i^* = 0$ or $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1 = 0$.

SVM with slack variables – soft margin SVM

Hard margin SVM

$$\min_{\boldsymbol{w}} \quad \boldsymbol{w}^T \boldsymbol{w}$$
$$\text{s.t.} \quad y_i(\boldsymbol{w}^T \boldsymbol{x}_i + w_0) \geq 1 \text{ for all } i$$



Soft margin SVM

$$\min_{\boldsymbol{w}, w_0} \quad \boldsymbol{w}^T \boldsymbol{w} + C \left(\sum_{i=1}^{n} \xi_i \right), \quad \text{where } C > 0,$$
$$\text{s.t.} \quad y_i(\boldsymbol{w}^T \boldsymbol{x}_i + w_0) \geq 1 - \xi_i \text{ for all } i, \ \xi_i \geq 0$$

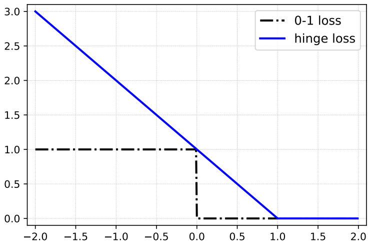
Loss function in soft-margin SVM

$$\min_{\boldsymbol{w}, w_0} \quad \boldsymbol{w}^T \boldsymbol{w} + C \left(\sum_{i=1}^{n} \xi_i \right), \quad \text{where } C > 0,$$
$$\text{s.t.} \quad y_i(\boldsymbol{w}^T \boldsymbol{x}_i + w_0) \geq 1 - \xi_i \;\text{ for all } i, \; \xi_i \geq 0$$

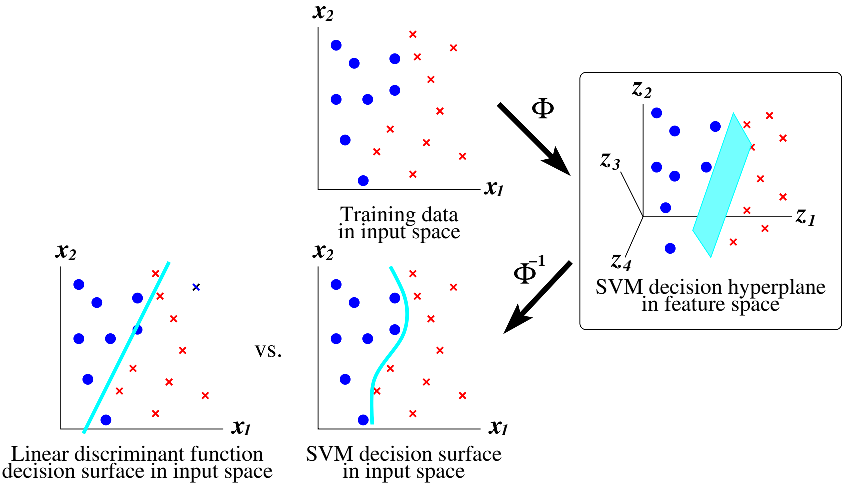
The hinge loss:

$$\ell(t) = \max(0, 1 - t) = \begin{cases} 0, & \text{if } t \geq 1, \\ 1 - t, & \text{otherwise,} \end{cases}$$

where $t = y(\mathbf{w}^T \boldsymbol{x} + w_0)$.



Non-linear SVM



Non-linear SVM (*cont.*)

- Conceptual steps to construct a non-linear SVM
  - Step 1 Transform $\boldsymbol{x}$ to $\phi(\boldsymbol{x})$ in a high-dimensional space (feature space)
  - Step 2 Train a SVM in the feature space
  - Step 3 Classify data in the feature space

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, y_i \, \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) + w_0$$

- Instead of applying the non-linear transformation and carrying out calculation in the feature space, use a kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ such that

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

(cf. '*kernel trick*')

$$L(\boldsymbol{\alpha}, \boldsymbol{\xi}) = -\frac{1}{2} \sum_{i,j=1}^{n} y_i \, y_j \, \alpha_i \, \alpha_j \, k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n} \alpha_i - C \sum_{i=1}^{n} \xi_i$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, y_i \, k(\boldsymbol{x}_i, \boldsymbol{x}) + w_0$$

Kernel functions for SVM

An example of kernel that maps data to a feature space explicitly

$$\begin{aligned}
k(\mathbf{a}, \mathbf{b}) &\triangleq (1 + \mathbf{a}^T \mathbf{b})^2 = (1 + a_1 b_1 + a_2 b_2)^2 \\
&= 1 + 2a_1 b_1 + 2a_2 b_2 + a_1^2 b_1^2 + 2a_1 b_1 a_2 b_2 + a_2^2 b_2^2 \\
&= (1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, \sqrt{2}a_1 a_2, a_2^2)(1, \sqrt{2}b_1, \sqrt{2}b_2, b_1^2, \sqrt{2}b_1 b_2, b_2^2)^T \\
&= \phi(\mathbf{a})^T \phi(\mathbf{b})
\end{aligned}$$

Popular kernels

Kernel	$k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
Polynomial	$(1 + \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^d$
Radial basis function (RBF)	$e^{-\gamma \lVert \boldsymbol{x}_i - \boldsymbol{x}_j \rVert^2}, \ \gamma > 0$
Hyperbolic tangent	$\tanh(\kappa_1 \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \kappa_2), \ \kappa_1 > 0, \kappa_2 < 0$

where $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ is an inner product (e.g. dot product) between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Making kernels

How can we ensure if a kernel works as an inner product in a feature space?

It should satisfy:

- $k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = \langle \phi(\mathbf{z}), \phi(\mathbf{x}) \rangle = k(\mathbf{z}, \mathbf{x})$
- $k(\mathbf{x}, \mathbf{z})^2 \leq k(\mathbf{x}, \mathbf{x})\, k(\mathbf{z}, \mathbf{z})$
- $K = (k(\mathbf{x}_i, \mathbf{x}_j))$, which is a n-by-n matrix, is positive semi-definite.

Mercer's theorem:
Suppose k is a continuous symmetric non-negative definite kernel, then k can be expressed as:

$$k(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z})$$

where $\{\phi_i\}$ are eigen-functions, $\|\phi_i\| = 1$, and $\{\lambda_i\}$ are positive eigenvalues $\lambda_i > 0$.

Making kernels from kernels

Letting k_1, k_2, and k_3 are kernels, we can create a new kernel k.

- $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$
- $k(\mathbf{x}, \mathbf{z}) = a k_1(\mathbf{x}, \mathbf{z}),\ a > 0$
- $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z})\, k_2(\mathbf{x}, \mathbf{z})$
- $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}) f(\mathbf{z})$
- $k(\mathbf{x}, \mathbf{z}) = k_3(\phi(\mathbf{x}), \phi(\mathbf{z}))$
- $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T B \mathbf{z},$ where B is a n-by-n matrix

Generalisation error of SVM *(NE)*

Assuming the class $\mathcal{F}$ of real-valued functions on the ball of radius R in $\mathbb{R}^n$ as

$$\mathcal{F} = \{\boldsymbol{x} \mapsto \mathbf{w} \cdot \boldsymbol{x} : \|\mathbf{w}\| \leq 1, \|\boldsymbol{x}\| \leq R\} .$$

If a classifier $\mathrm{sgn}(f) \in \mathrm{sgn}(\mathcal{F})$ has margin at least γ on all the training examples, with probability at least $1 - \delta$ over n random examples, f has error no more than

$$L_D(f) \leq \frac{k}{n} + \sqrt{\frac{c}{n}\left(\frac{R^2}{\gamma^2}\log^2 n + \log\left(\frac{1}{\delta}\right)\right)}$$

where k is the number of labelled training examples with margin less than γ, c is a constant,

$$\text{VC-dim}(f) \leq \min(\frac{R^2}{\gamma^2}, n) + 1$$

Experiments on US Postal Service Database

C. Cortes and V. Vapnik, "Support-Vector Networks", Machine Learning 20, 273–297 (1995). https://doi.org/10.1007/BF00994018

US Postal Service Database (handwritten digits):

Training samples	*7300*
Test samples	*2000*
Image resolution	16×16 *pixels*

Classifier	Err. [%]
Human performance	2.5
Decision tree, CART	17.0
Decision tree, V4.5	16.0
Best 2 layer NN	6.6
LeNet1 (5 layers)	5.1

d	Err. [%]	Support vectors	Dimensionality of feature space
1	12.0	200	256
2	4.7	127	~ 33000
3	4.4	148	$\sim 1 \times 10^{6}$
4	4.3	165	$\sim 1 \times 10^{9}$
5	4.3	175	$\sim 1 \times 10^{12}$
6	4.2	185	$\sim 1 \times 10^{14}$
7	4.3	190	$\sim 1 \times 10^{16}$

d: degree of polynomial kernel

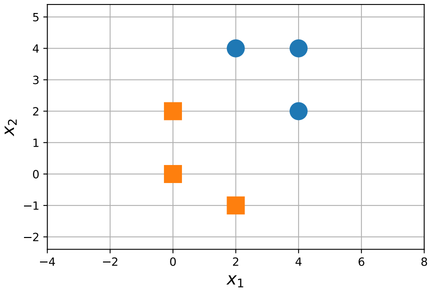
Some notes on SVMs

- How to find w_0? $\cdots$ use $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) = 1$ for support vectors
- How to choose the regulariser C? $\cdots$ use a validation set
- How to solve the constrained quadratic optimisation problem in SVM practically? It requires a kernel matrix of n-by-n.
  - Gradient, sub-gradient, coordinate ascent/descent
  - Sequential Minimal Optimisation (SMO) [John Platt, 1998]
  - LIBSVM [Chih-Chung Chang and Chih-Jen Lin]: a SVM software tool with SMO
- How to apply SVMs to multi-class classification problems?
- Performance deterioration (NB: not very specific to SVMs)
  - Heavily-overlapped data sets
  - Imbalanced data sets
  - (Too many support vectors)
  - (Large data sets)
- Output interpretability

Quizzes

Consider a SVM with a linear kernel run on the following data set.

x_1	x_2	y
2.0	4.0	1
4.0	2.0	1
4.0	4.0	1
0.0	2.0	2
2.0	-1.0	2
0.0	0.0	2



1. Using your intuition, what weight vector do you think will result from training an SVM on this data set?
2. Plot the data and the decision boundary of the weight vector you have chosen.
3. Which are the support vectors? What is the margin of this classifier?