Machine Learning in Healthcare

#L10-Support vector machines

Technion-IIT, Haifa, Israel

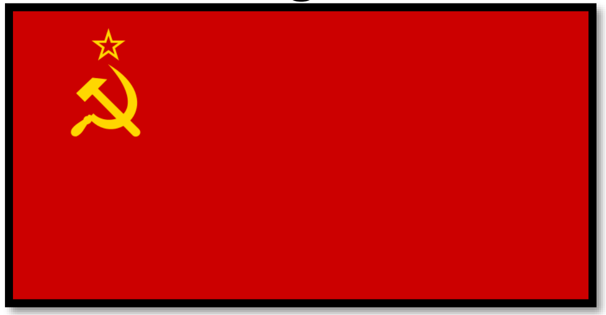
Asst. Prof. Joachim Behar
Biomedical Engineering Faculty, Technion-IIT
Artificial intelligence in medicine laboratory (AIMLab.)
https://aim-lab.github.io/
Twitter: @lab_aim

The inventor of SVM

Codename
Vladimir Vapnik

Special power
Machine learner!

Place of origin



SVM

Vladimir Vapnik, Corinna Cortes. "Support vector networks," Machine Learning, vol. 20, pp. 273-297, 1995.

$y \in \{-1, 1\}$

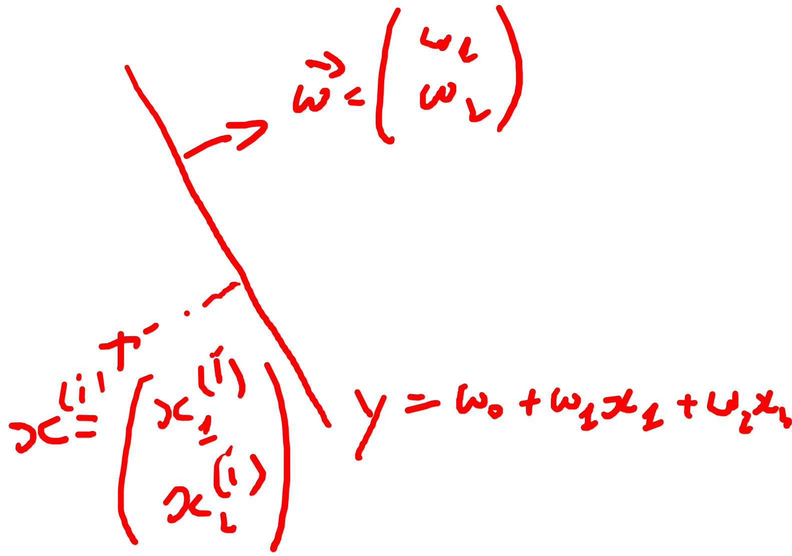
$h(x) \geq 0$ for $y = +1$

$h(x) \leq 0$ for $y = -1$

- distance between a point line / hyperplane :

$$d = \frac{|w_1 x_1 + w_2 x_2 + w_0|}{\|w\|}$$

$$\boxed{d = \frac{|w^T x^{(i)} + b|}{\|w\|}}$$



$$y^{(i)} h(x^{(i)}) \geq 0$$

$y^{(i)}$	$h(x^{(i)})$
$+1$	≥ 0
-1	≤ 0

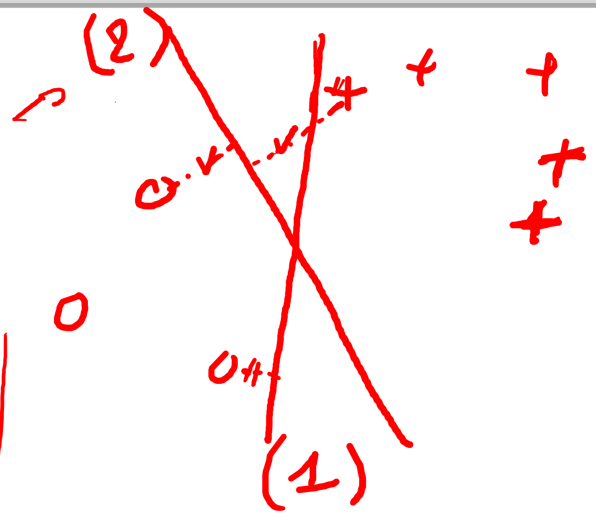
$$d^{(i)} = \frac{y^{(i)} h(x^{(i)})}{\|w\|} = \boxed{\frac{y^{(i)}(w^T x^{(i)} + b)}{\|w\|}}$$

- move to a new space

$$\boxed{d^{(i)} = \frac{y^{(i)}(w^T \phi(x^{(i)}) + b)}{\|w\|}}$$

4. maximum margin

$$\min_{i\in[\![1,m]\!]} \left(\frac{y^{(i)}\left(w^T\phi(x^{(i)})+b\right)}{\|w\|} \right)$$



$d_1 < d_2$

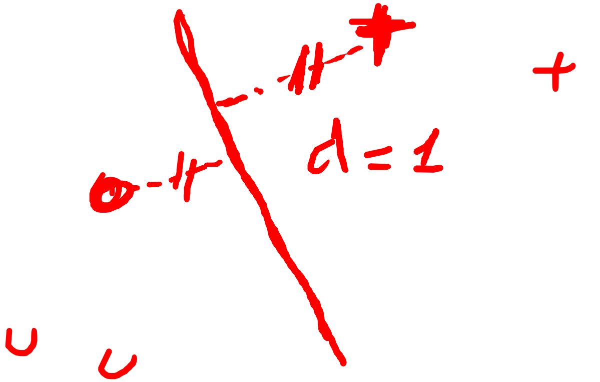
$$\boxed{(w^*, b^*) = \underset{w,b}{\arg\max} \left(\frac{1}{\|w\|} \min_{i\in[\![1,m]\!]} \left(y^{(i)}\left(w^T\phi(x^{(i)})+b\right)\right)\right)}$$

$$\Updownarrow$$

$y^{(i)}\left(w^T\phi(x^{(i)})+b\right) = 1 \leftarrow$ i = closest

nb of examples

$y^{(i)}\left(w^T\phi(x^{(i)})+b\right) \geq 1, \forall i \in [\![1, m]\!]$



$$\begin{cases} (w^*, b^*) = \arg\max \left(\dfrac{1}{\|w\|}\right) \\ y^{(i)}\left(w^T\phi(x^{(i)})+b\right) \geq 1, \forall i \in [\![1, m]\!] \end{cases}$$

$$\begin{cases} \underset{w,b}{\text{argmin}} \left\{ \frac{1}{2} \|w\|^2 \right\} \\ y^{(i)} \left(w^T \phi(x^{(i)}) + b \right) \geq 1 \end{cases}$$

- Lagrange multiplier

$$\mathcal{L}(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{m} a_i \left\{ y^{(i)} \left(w^T \phi(x^{(i)}) + b \right) - 1 \right\}$$

$$\begin{cases} w = \sum_{i=1}^{m} a_i y^{(i)} \phi(x^{(i)}) \\ 0 = \sum_{i=1}^{m} a_i y^{(i)} \end{cases}$$

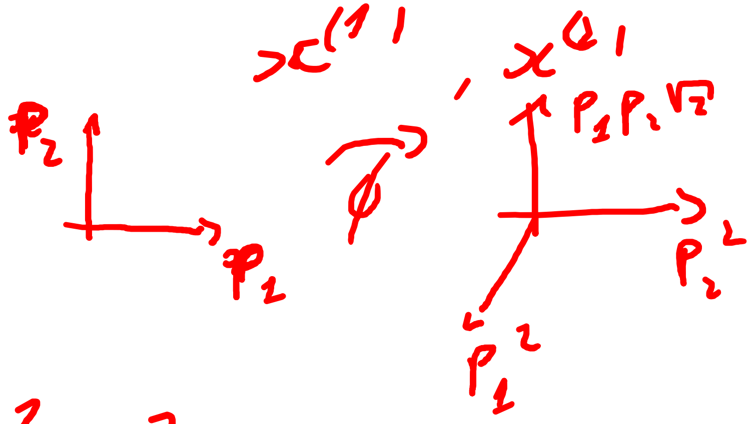
- Dual Lagrange multiplier

$$\mathcal{L}_{dual}(a) = \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j y^{(i)} y^{(j)} \phi(x^{(i)}) \phi(x^{(j)})$$

$$\boxed{k(x^{(i)}, x^{(j)})}$$

$$\left(x^{(i)T} x^{(j)} \right)^2$$

- The kernel trick



$$k(x^T, z^T) = (x^T z)^2$$

$$= (x_1 z_1 + x_2 z_2)^2$$

$$\begin{pmatrix} BP_x \\ HR_x \end{pmatrix} \cdot \begin{pmatrix} BP_2 \\ HR_2 \end{pmatrix} \quad \begin{pmatrix} BP^2 \\ \sqrt{2}\, BP\, HR \\ HR^2 \end{pmatrix}$$

$$= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2$$

$$= \underbrace{\begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}}_{\phi(x)} \begin{array}{c} 1 \\ 2 \\ 1 \end{array} \cdot \underbrace{\begin{pmatrix} z_1^2 \\ \sqrt{2} z_1 z_2 \\ z_2^2 \end{pmatrix}}_{\phi(z)}$$

Original:

$(x^T z)^2$

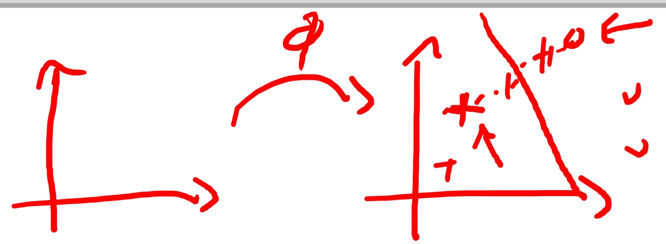
4 operations

$$\phi : x \longrightarrow \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}$$

New:

$\phi(x)\phi(z)$

$8 + 3 = 11$ operations.



(1) SVM "widest street" / max margin problem.

(2) distance from a point to plane.

(3) optimisation pb

(4) Lagrange multiplier

(5) "kernel trick"

(6) new prediction

$$h(x) = w^T \phi(x) + b \qquad w = \sum_{i=1}^{m} a_i \, y^{(i)} \phi(x^{(i)})$$

$$h(x) = \sum_{i=1}^{m} a_i \, y^{(i)} \underbrace{\phi(x^{(i)}) \phi(x)}_{k(x^{(i)}, x)} + b$$

$x^{(i)}$, $\forall i \in [\![1, m]\!]$

x

a_i non-zero

- Kernels:

  Linear : $k(x, z) = (x^T z)$

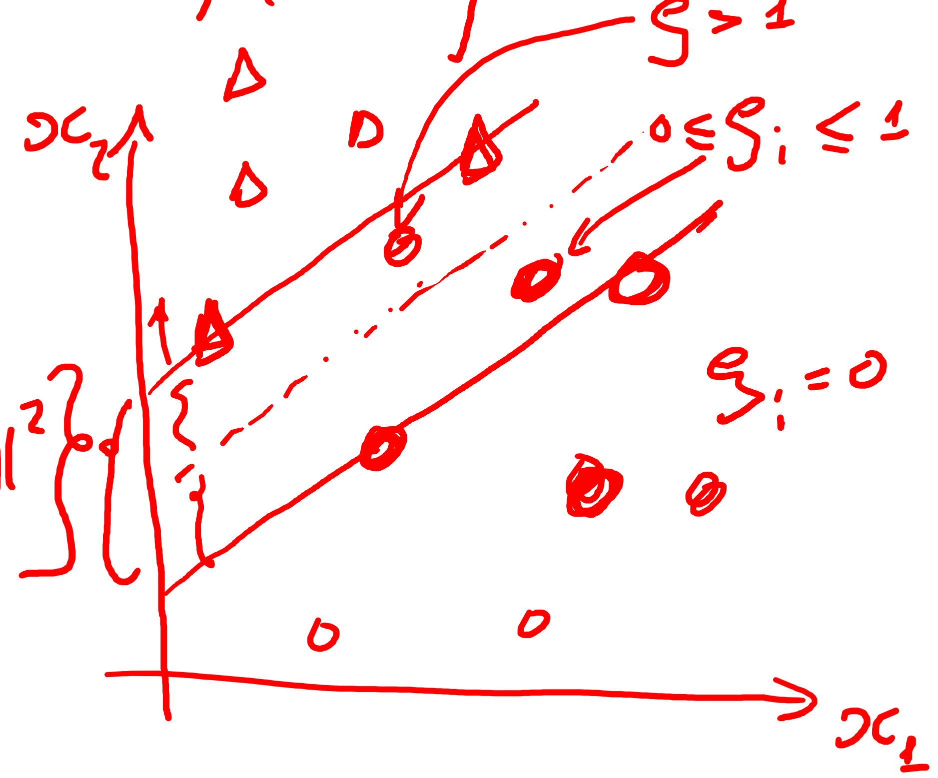
  Polynomial : $k(x,z) = (1 + x^T z)^d$

  Gaussian : $k(x,z) = \exp\left(-\|x-z\|^2 / (2\sigma^2)\right)$ RBF

- Slack variable : ξ_i

  $y^{(i)} h(x^{(i)}) \geq 1 - \xi_i$

$$(w^*, b^*) = \underset{b,w}{\operatorname{argmin}} \left\{ \underbrace{C \sum_{i=1}^{m} \xi_i}_{\text{Capacity}} + \frac{1}{2}\|w\|^2 \right\}$$



- hinge error :

$$E_{SVM}(x^{(i)}) = \max\left(0,\ 1 - y^{(i)} h(x^{(i)})\right)$$
$$= \big(1 - \underbrace{y^{(i)} h(x^{(i)})}_{z}\big)_+$$

- if $h(x^{(i)}) \geq 1$ and $y^{(i)} = 1$

  then $h(x^{(i)}) y^{(i)} \geq 1$

  $E(z) = 0$

- if $h(x^{(i)}) \leq -1$ but $y^{(i)} = 1$

  then $h(x^{(i)}) y^{(i)} \leq -1$

  $z = 1 - h(x^{(i)}) y^{(i)} \geq 2$

- cost function

$$J_{svm} = \sum_{i=1}^{m} E_{svm}(x^{(i)}) + \lambda \|w\|^2$$

$$= \underbrace{\sum_{i=1}^{m} \max(0, 1 - y^{(i)} h(x^{(i)}))}_{} + \lambda \|w\|^2$$

$$\boxed{J_{svm} = C \sum_{i=1}^{m} \max(0, 1 - y^{(i)} h(x^{(i)})) + \|w\|^2}$$

Reminder

$$(w^*, b^*) = \underset{b,w}{\operatorname{argmin}} \left\{ C \sum_{i=1}^{m} \xi_i + \frac{1}{2}\|w\|^2 \right\}$$

AIMLab.

- LR: $J_{LR}(w) = \sum_{i=1}^{m} \log\left(1 + \exp\left(-w^T x^{(i)} \cancel{y}^{(i)}\right)\right) + \underbrace{\cancel{\lambda}\|w\|^2}$
- SVM: $J_{SVM}(w) = \cancel{C} \sum_{i=1}^{m} \max\left(0, 1 - y^{(i)} h_w(x^{(i)})\right) + \underbrace{\|w\|^2}$

$\text{hyper} = \{C, \cancel{\gamma}\}$

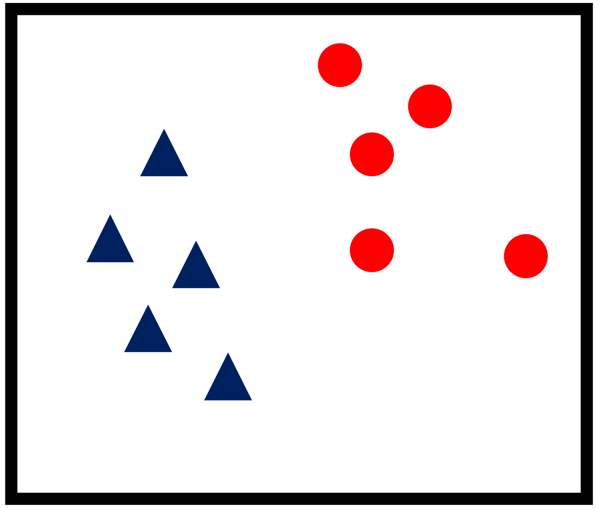
Introduction

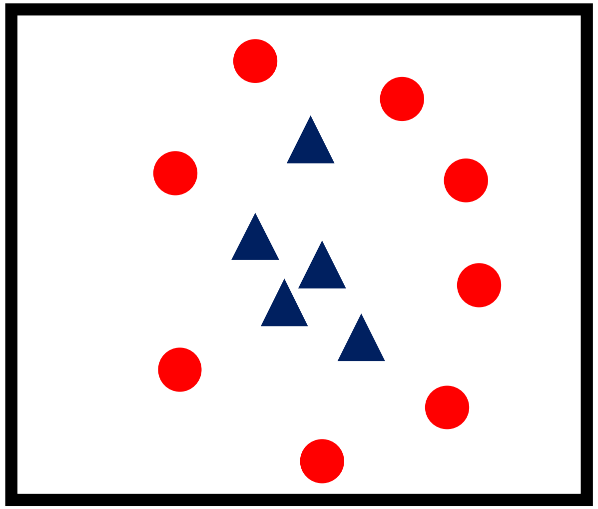
Two class classification

- We have a training set consisting of:
  - m examples: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
  - With target labels: $\{y^{(1)}, y^{(2)}, \dots, y^{(m)}\}$
- We wish to learn $h(x)$ so that:
  - $h(x^{(i)}) = \begin{cases} \geq 0 & for\ \ y^{(i)} = +1 \\ < 0 & for\ \ y^{(i)} = -1 \end{cases}$

▲Versicolor

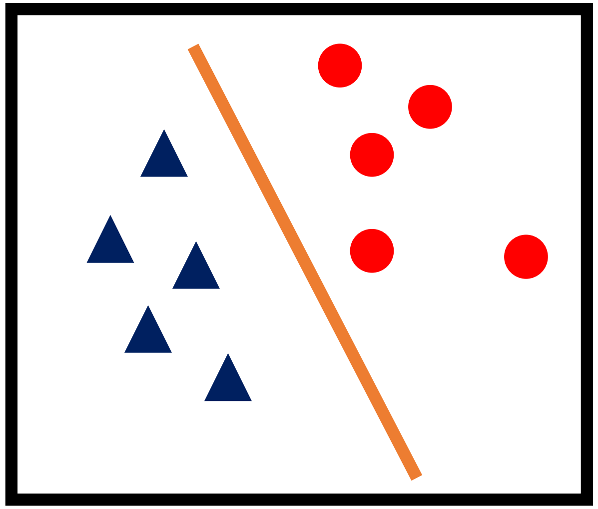
● Virginica

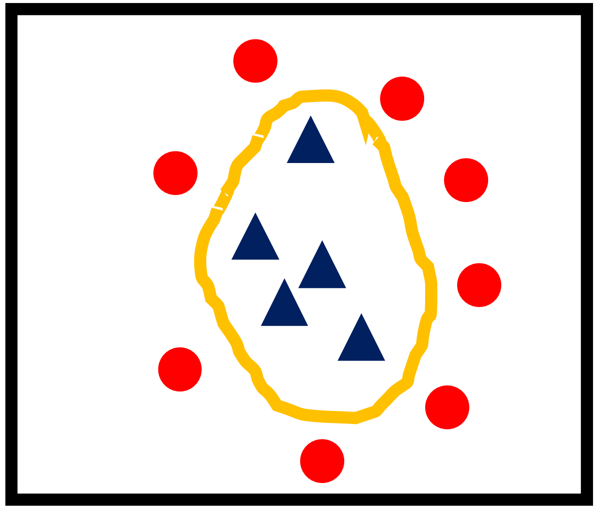




Linear versus non-linear

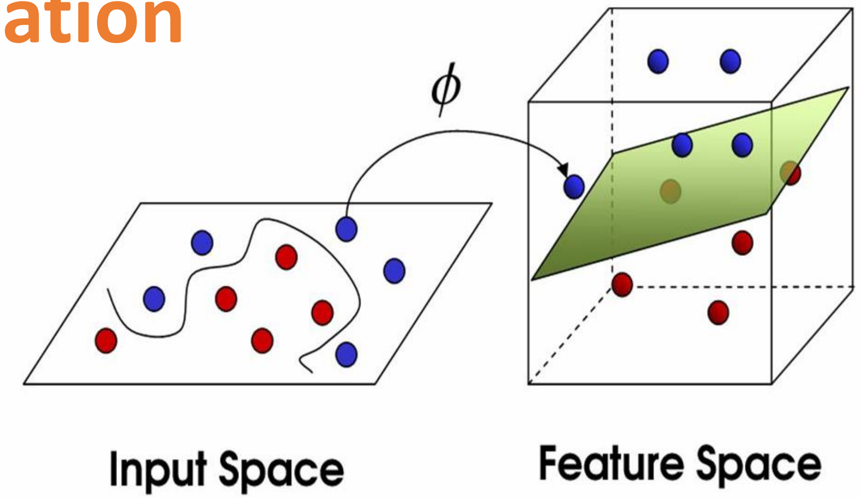
- Linear model
  - $z = w^T x + b$
  - $h(x) = \sigma(z)$
- Non-linear model
  - ?

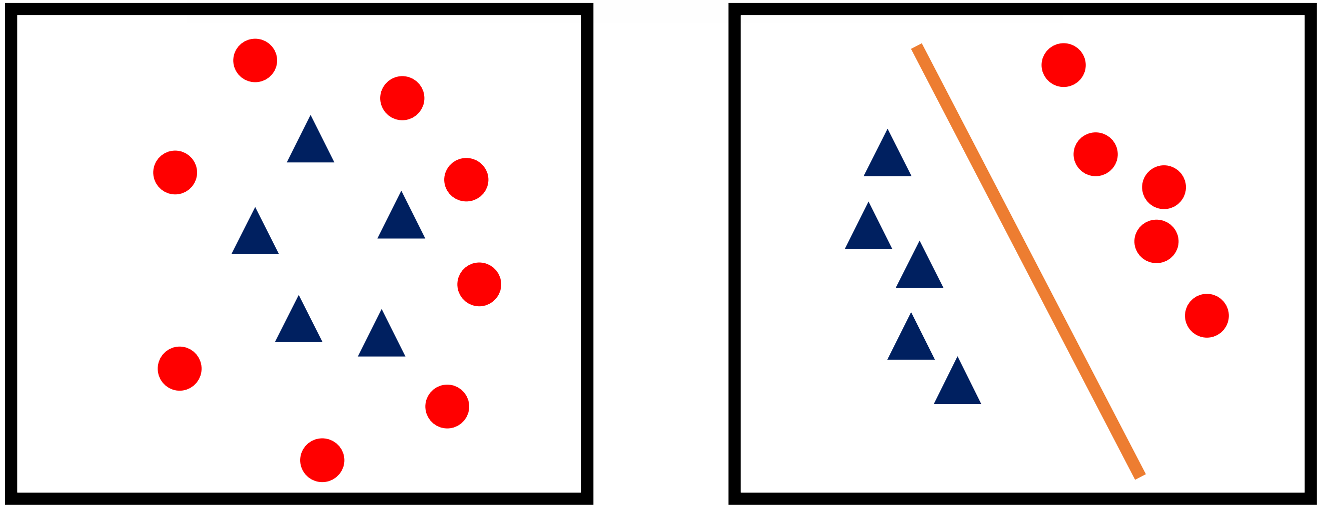




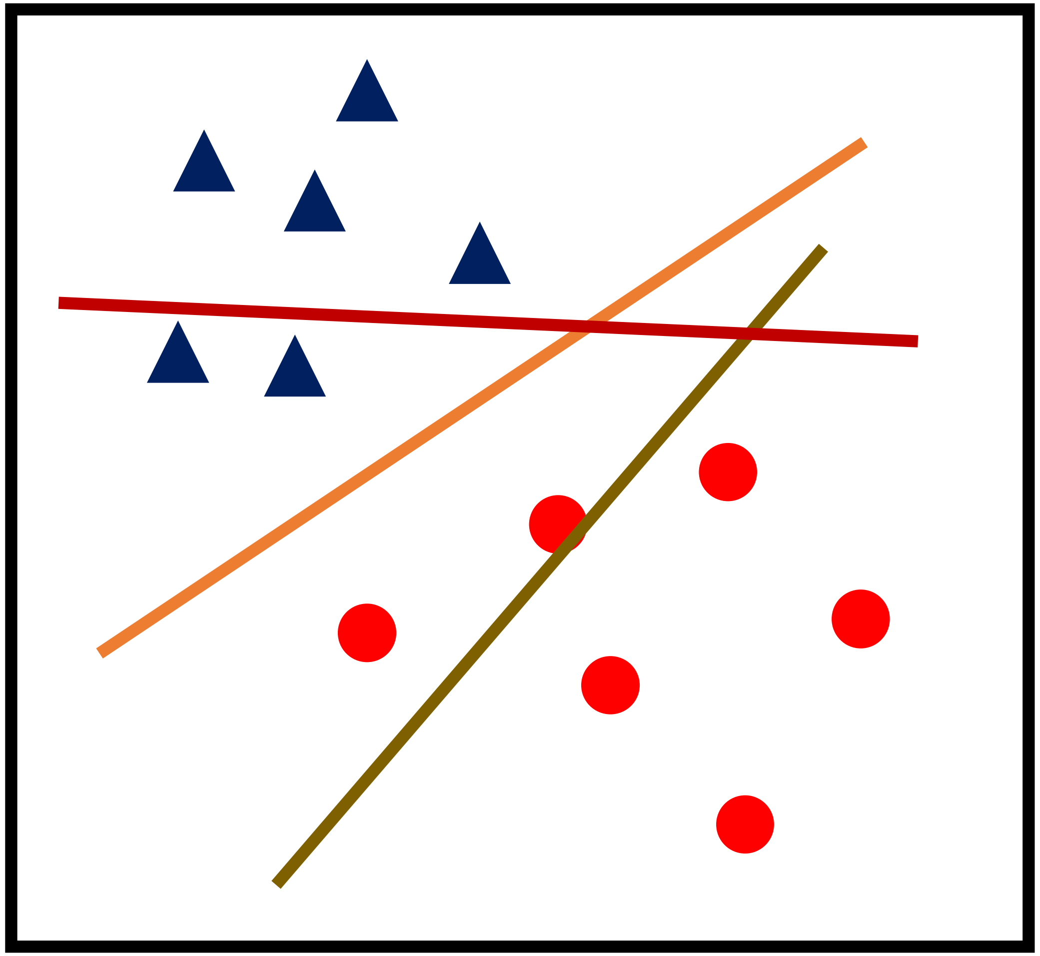
Feature space transformation

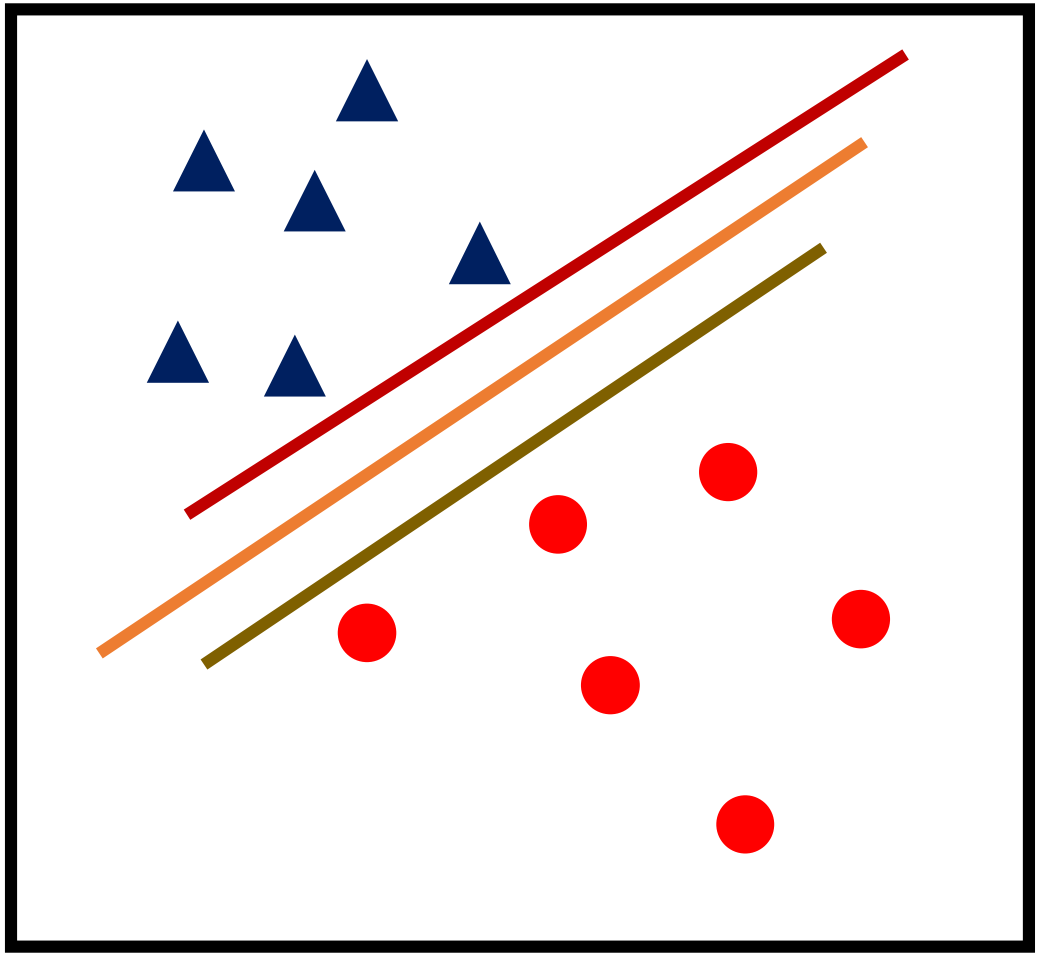
- $h(x) = w^T \phi(x) + b$





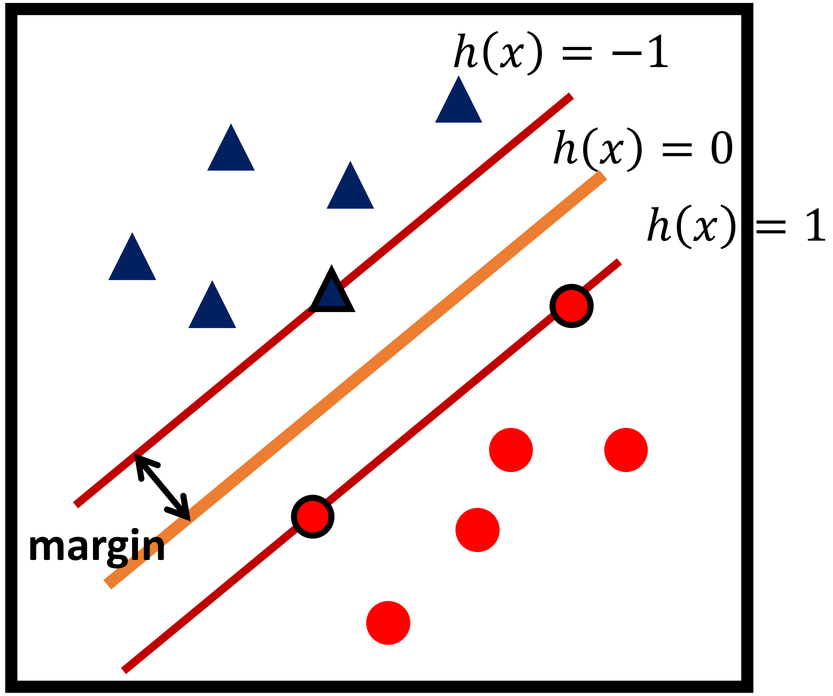
SVM - Intuition



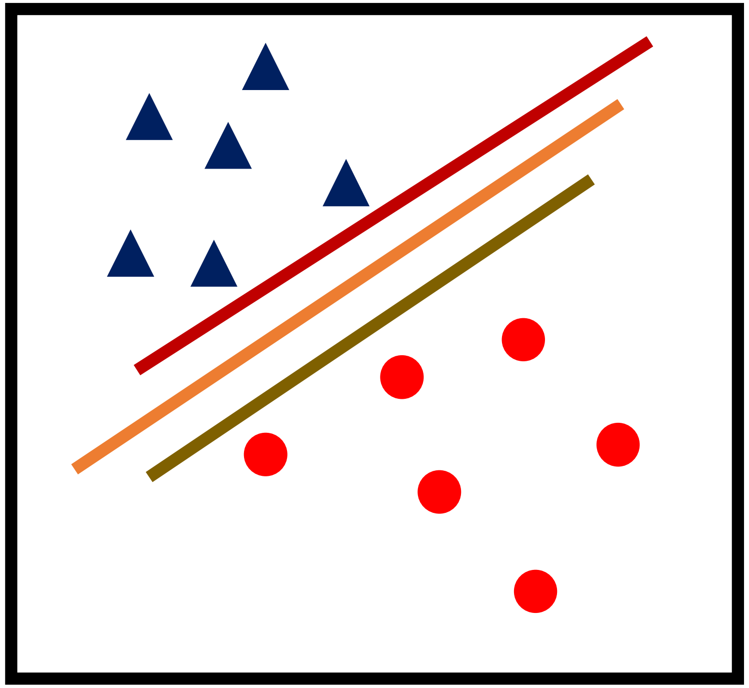


SVM – Intuition, Maximum margin

- In support vector machines the decision boundary is chosen to be the one for which the margin is maximised.



What is the most likely margin?



SVM - Derivation

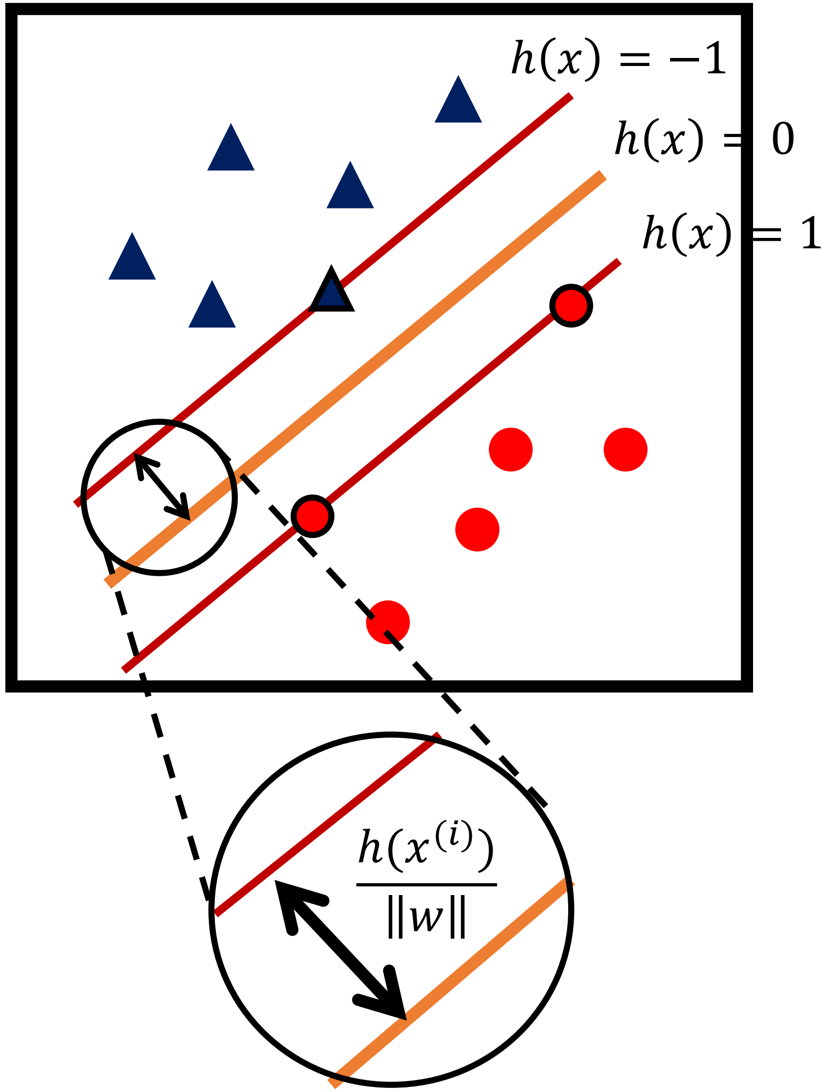
AIMLab.

Derivation

- Model: $h(x) = w^T \phi(x) + b$
- Defining margin:
  - $\frac{y^{(i)} h(x^{(i)})}{\|w\|} = \frac{y^{(i)}(w^T \phi(x^{(i)}) + b)}{\|w\|}$
- Maximum margin classifiers:

$$\underbrace{argmax_{w,b} \left\{ \frac{1}{\|w\|} \overbrace{min_i [y^{(i)}(w^T \phi(x^{(i)}) + b)]}^{\text{Distance to the closest point}} \right\}}_{\text{Maximise distance}}$$

$$\Leftrightarrow \quad argmin_{w,b} \left\{ \frac{1}{2} \|w\|^2 \right\}$$
$$y^{(i)}(w^T \phi(x^{(i)}) + b) \geq 1$$



Dual representation of the maximum margin problem

- To solve the constrained optimization problem we introduce the Lagrange multipliers $a_n \geq 0$ with one multiplier for each of the constraints. The Lagrangien function:
  - $L(w, b, a) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} a_i\{y^{(i)}\left(w^T\phi\left(x^{(i)}\right) + b\right) - 1\}$
  - $a_i \geq 0$
- The minus sign in front of the Lagrange multiplier is because we minimize with respect to w and b and maximize with respect to a.
- This is a **quadratic optimization problem** that we could solve using gradient descent. But instead we look at the **Dual representation of the maximum margin problem**.

Dual representation of the maximum margin problem

- Setting the derivative with respect to w and b equal to zero:
  - $w = \sum_{i=1}^{m} a_i y^{(i)} \phi(x^{(i)})$
  - $0 = \sum_{i=1}^{m} a_i y^{(i)}$
- If we inject that back in the Lagrange multiplier…

Derivation

- Dual representation of the maximum margin problem
  - $\tilde{L}(a) = \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j y^{(i)} y^{(j)} \boxed{\phi(x^{(i)})^T \phi(x^{(j)})}$
  - Under the constraints:
    - $a_i \geq 0,\ \ i \in [\![1..m]\!]$
    - $\sum_{i=1}^{m} a_i y^{(i)} = 0$
    - $w = \sum_{i=1}^{m} a_i y^{(i)} \phi(x^{(i)})$

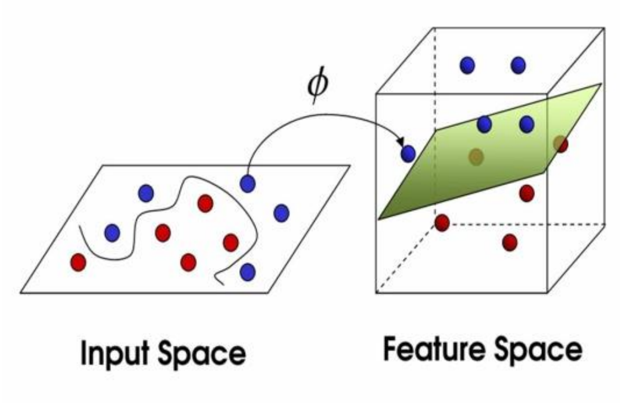
We did all that work for that. Why?

- We write $k\left(x^{(i)}, x^{(j)}\right) = \phi(x^{(i)})^T \phi(x^{(j)})$ the kernel function.

The Kernel trick

$$k\left(x^{(i)}, x^{(j)}\right) = \phi(x^{(i)})^T \phi(x^{(j)})$$

Remember



But sometime ϕ can map the input space to a high dimension one...

...this is computationaly costly

This is where the kernel plays magic!

AIMLab.

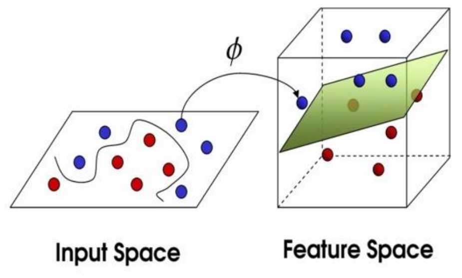
The Kernel trick

$$
\begin{aligned}
k(\mathbf{x}^T, \mathbf{z}^T) &= (\mathbf{x}^T\mathbf{z})^2 \\
&= (x_1 z_1 + x_2 z_2)^2 \\
&= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\
&= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)(z_1^2, \sqrt{2}z_1 z_2, z_2^2)^T \\
&= \phi(\mathbf{x})^T \phi(\mathbf{z})
\end{aligned}
$$

Moving from 2-D to 3-D space

4 operations

$$(\mathbf{x}^T\mathbf{z})^2$$



11 operations

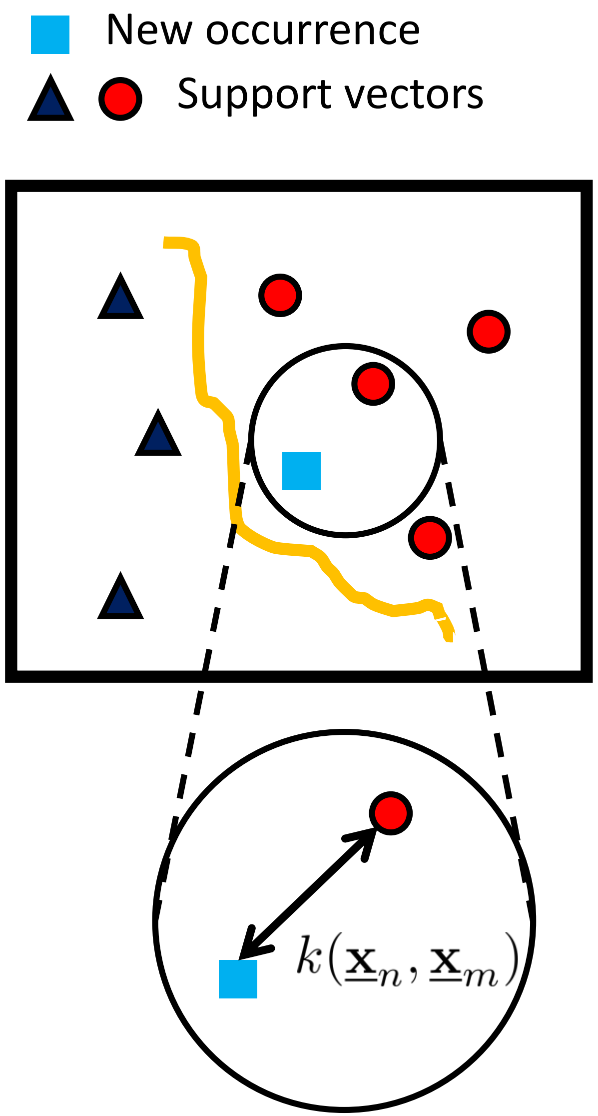
$$\phi(\mathbf{x})^T \phi(\mathbf{z})$$

With Kernel functions we do not need to move to the new feature space explicitly! This saves computational cost.

Kernel

- Kernel $k(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)})$,
- A new example x is classified by computing:
  - $h(x) = \sum_{i=1}^{m} a_i y^{(i)} k(x, x^{(i)}) + b$

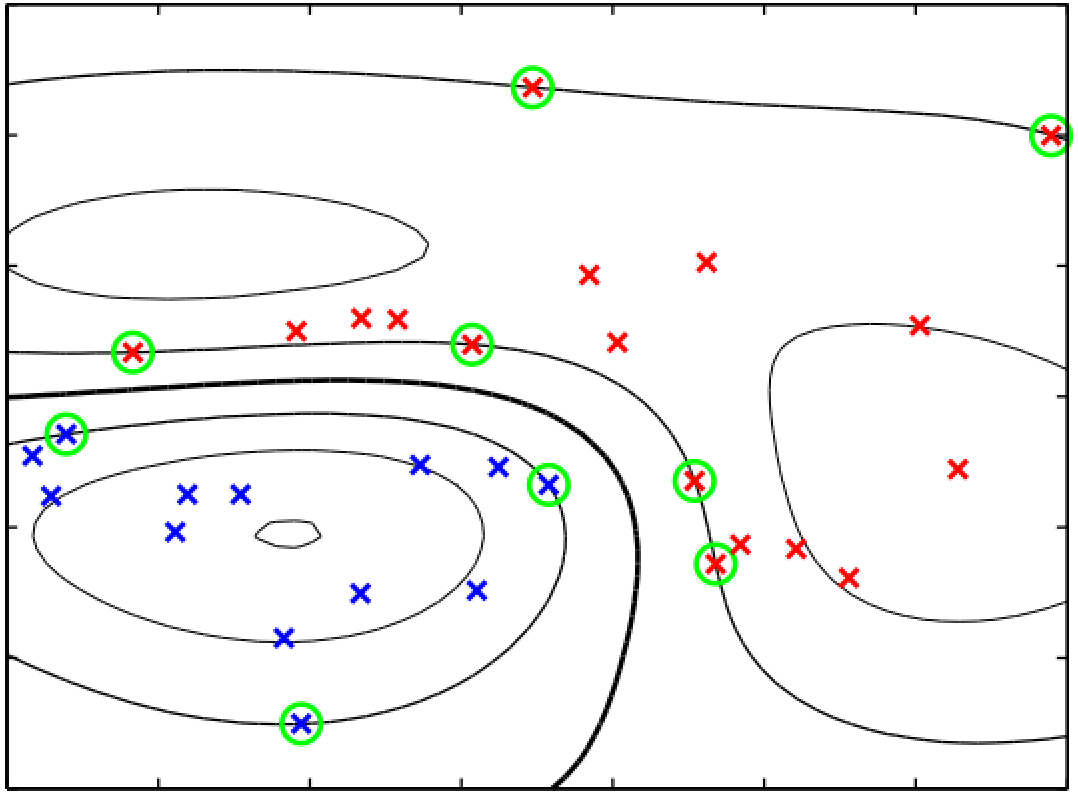
Kernel method = training data points or a <u>subset</u> of them is used during the prediction phase.



AIMLab.

Sparsity

- $h(x) = \sum_{i=1}^{m} a_i y^{(i)} k\left(x, x^{(i)}\right) + b$
- Only the support vectors are kept for the prediction. Thus the prediction relies on a limited set of vectors – this is sparsity!



Bishop Fig 7.2

SVM has a sparse solution, this means that the predictions for new inputs depends only on the kernel function evaluated on a subset of the training data points.

AIMLab.

Classification

Examples of kernels

Linear

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})$$

Polynomial – all terms up to degree *d*

$$k(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^d$$

Gaussian (also called RBF)– Infinite dimensional feature space

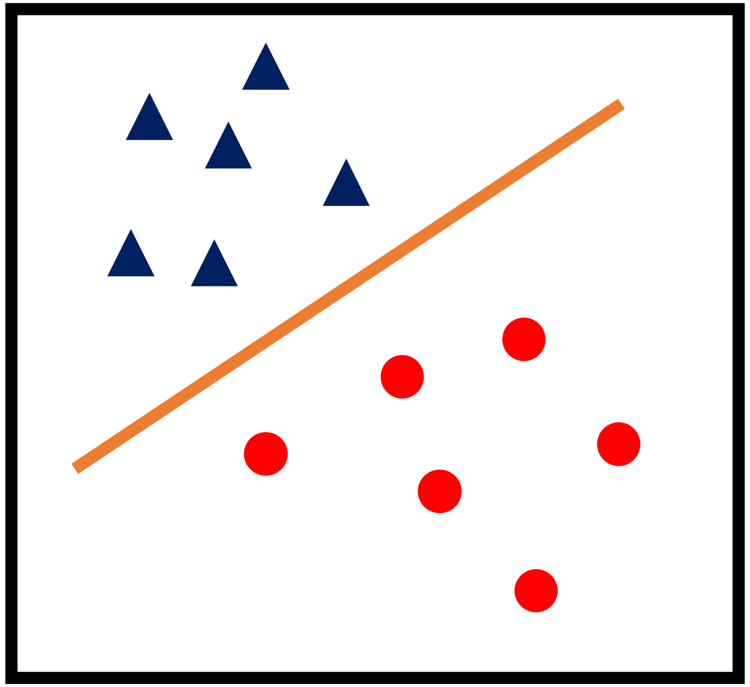
$$k(\mathbf{x}, \mathbf{z}) = exp(-||\mathbf{x} - \mathbf{z}||^2/(2\sigma^2))$$

Overlapping class distribution

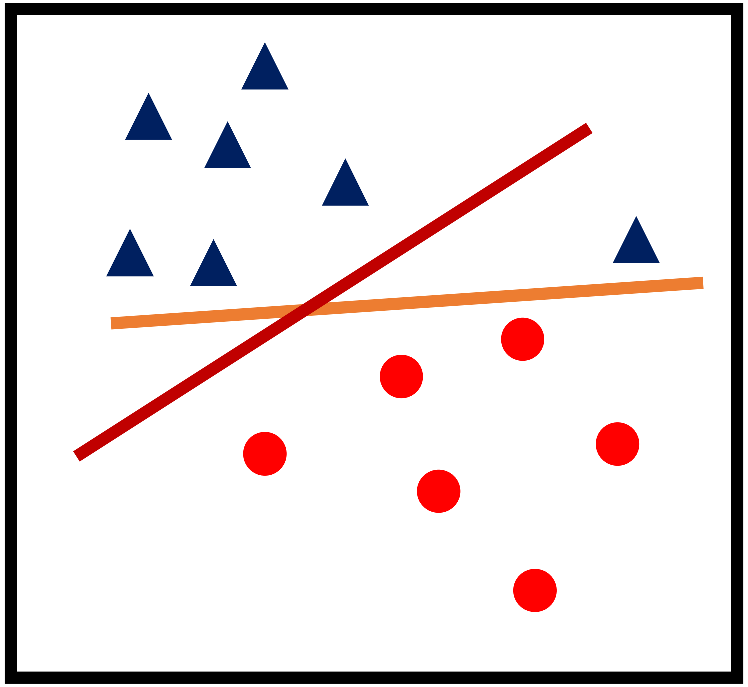
Classification

Overlapping class distribution

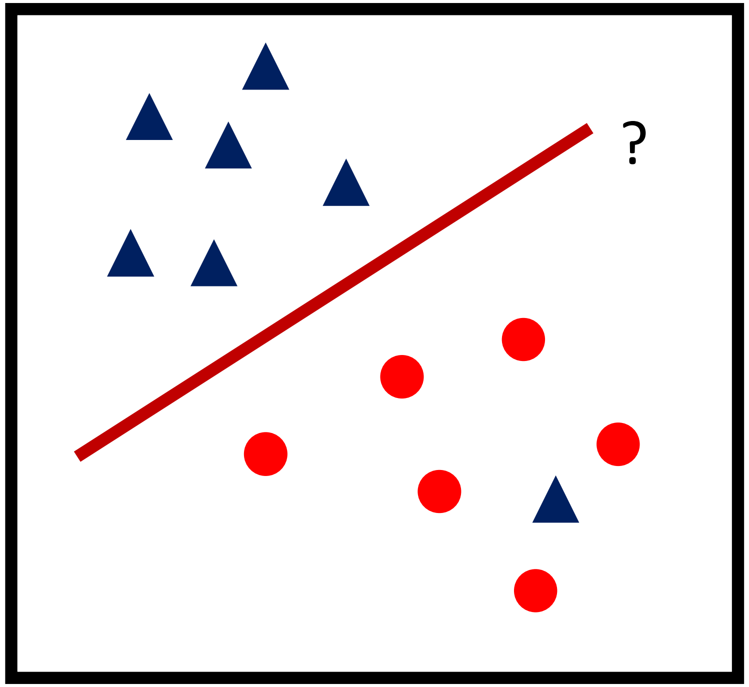
Ideal life



More real!

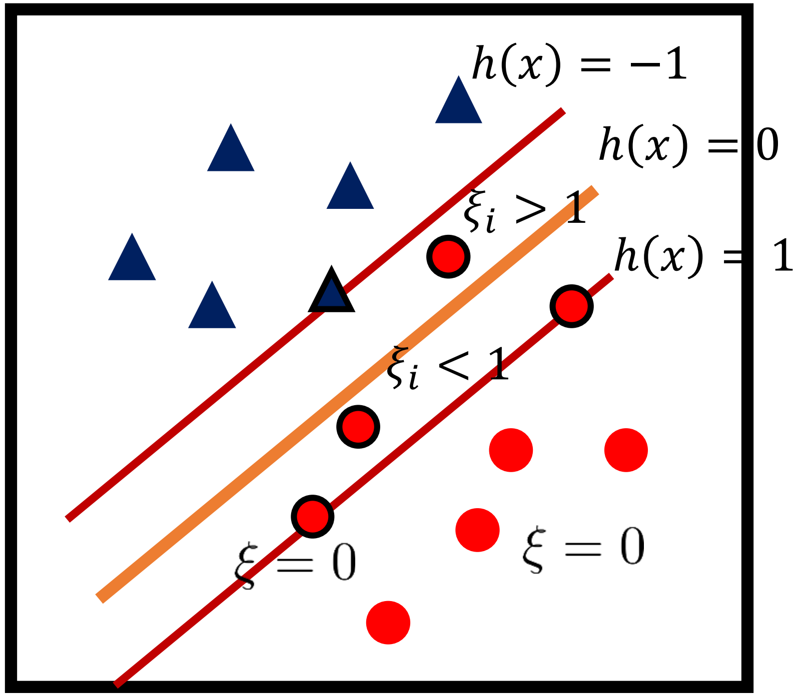


Confrontation with reality!



Overlapping class distribution

- The mathematical problem was formulated as:
  - $argmin_{\,w,b} \left\{\frac{1}{2}\|w\|^2\right\}$
  - $y^{(i)}\left(w^T\phi\left(x^{(i)}\right)+b\right) \geq 1$
- The mathematical problem can be re-formulated as:
  - $argmin_{\,w,b} \left\{C\sum_{i=1}^{m}\xi_i + \frac{1}{2}\|w\|^2\right\}$
  - $y^{(i)}\left(w^T\phi\left(x^{(i)}\right)+b\right) \geq 1-\xi_i, \xi \geq 0$
  - ξ is called the "slack variable".



Penalise misclassification

How much do you penalise

Hyperparameters

AIMLab.

Capacity and Gaussian kernel

- Hyperparameters:
  - $argmin_{w,b}\left\{C\sum_{i=1}^{m}\xi_i+\frac{1}{2}\|w\|^2\right\}$

How much do you penalise

  - $k(x,z)=\exp(-\|x-z\|^2\gamma)$
    - $\gamma=1/(2\sigma^2)$

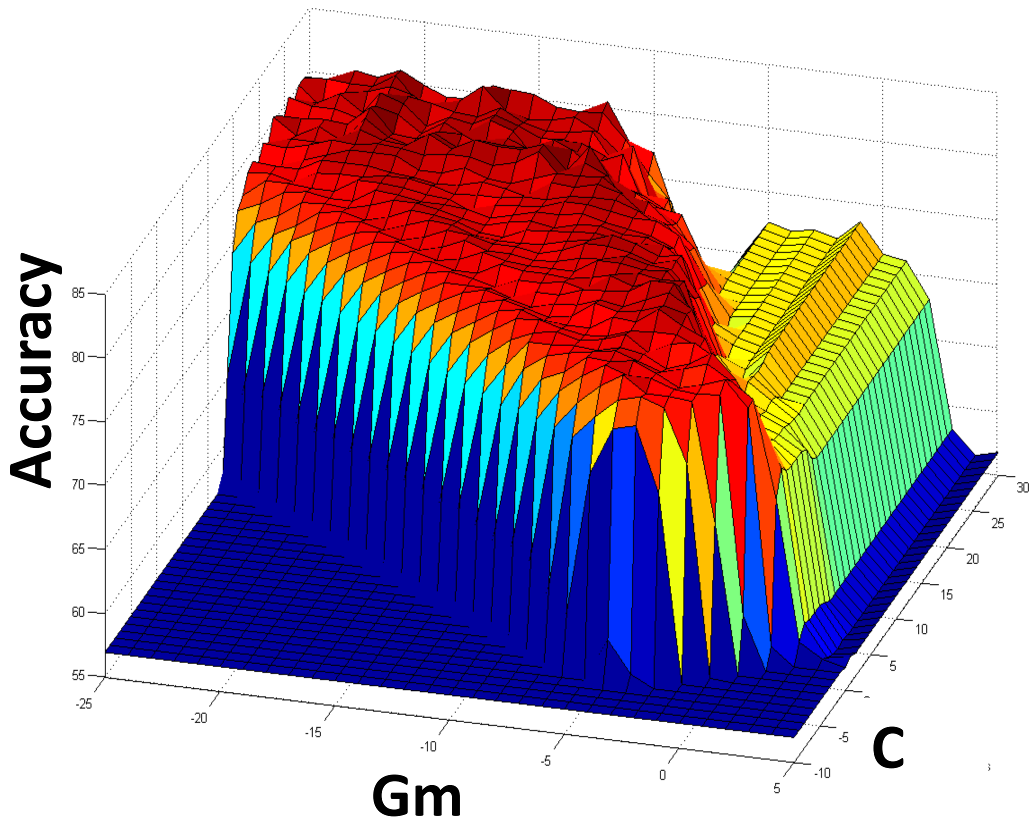
How much we fit the training data

- Gamma, kernel coefficient for RBF. The higher Gamma the more we fit the training data. So too high of a Gamma can cause overfitting.

Grid search

$$k(x,z) = \exp(-\|x-z\|^2\gamma)$$

$$argmin_{w,b}\left\{C\sum_{i=1}^{m}\xi_i + \frac{1}{2}\|w\|^2\right\}$$



- Using **cross-validation** to find a good combination of theses two hyper parameters and avoid **overfitting**.
- Set of parameters with the best cross validation accuracy is chosen
- Search typically performed for exponentially growing sequences ($C=2^{-5}, 2^{-4}, \ldots, 2^{15}$ and $Gm=2^{-15}, 2^{-14}, \ldots, 2^{3}$)

Behar, Joachim, et al. "SleepAp: an automated obstructive sleep apnoea screening application for smartphones." *IEEE journal of biomedical and health informatics* 19.1 (2015): 325-331.

Hyper-parameters

Random search

- Grid search is computationally expensive.
- Particularly when the number of parameters to tune is high and when you have a large training set.
- An alternative is to use **Random search**[1]

See toolbox: http://physionet.org/physiotools/random-search/

Contributed by Alistair Johnson and Joachim Behar.

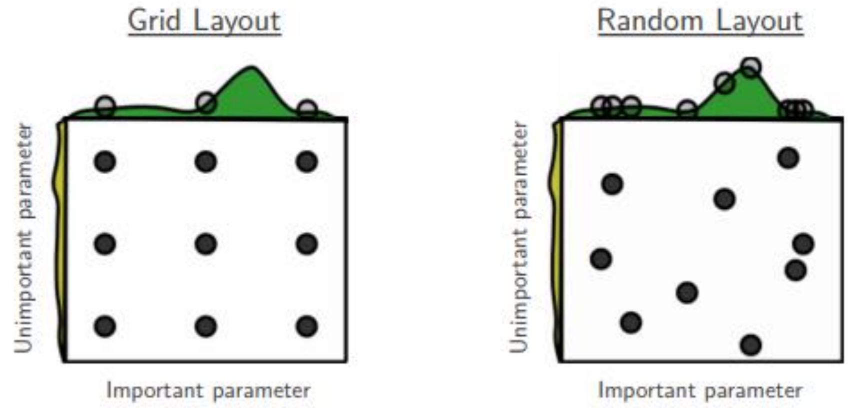
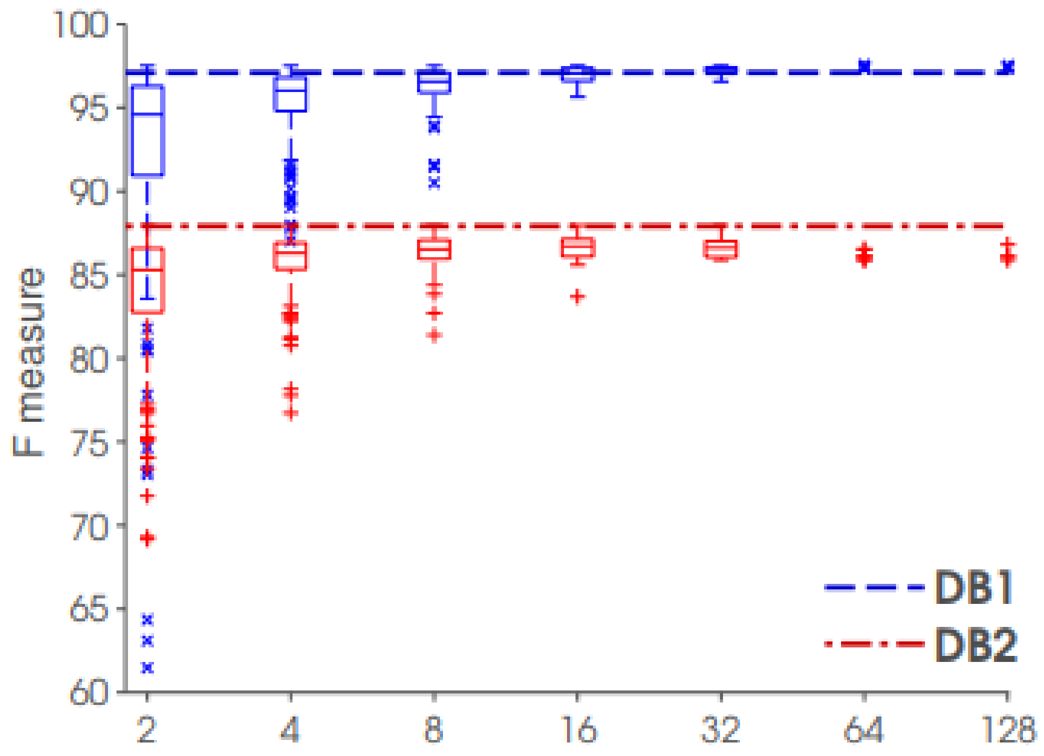


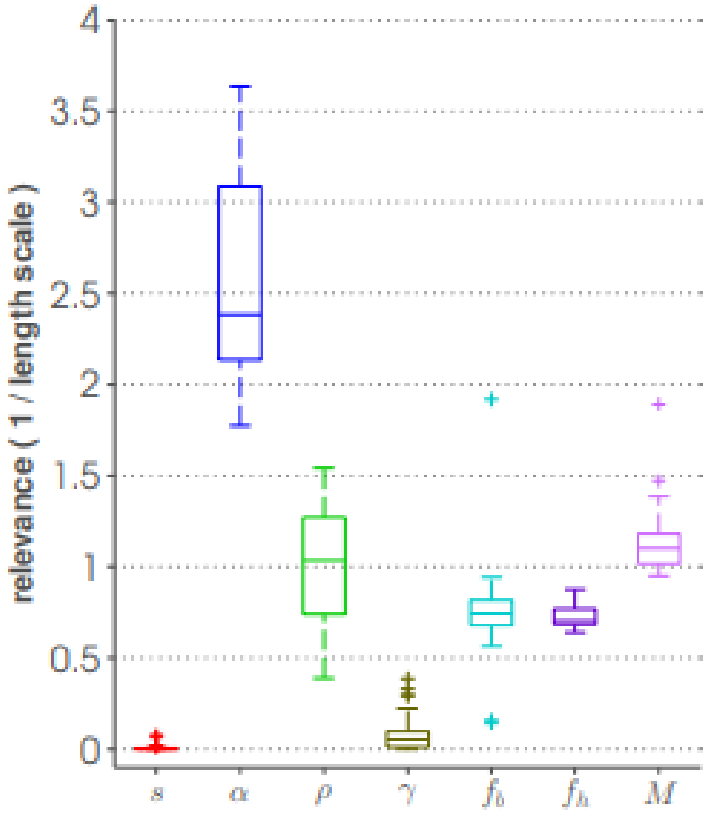
Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

[1] Bergstra, James, and Yoshua Bengio. "*Random search for hyper-parameter optimization.*" The Journal of Machine Learning Research 13 (2012): 281-305.

Hyper-parameters

Random search





Behar, Joachim, et al. "An echo state neural network for foetal ECG extraction optimised by random search." *Proc. Adv. Neural Inf. Process. Syst* (2013).

SVM and Probabilities

- SVM does not provide probabilistic outputs. Rather SVM make a decision for a new occurrence.
- What if we want a probability estimate?
- We can fit a logistic sigmoid to the outputs of the trained SVM

$$P(t = 1|\mathbf{x}) = \sigma(Ay(\mathbf{x}) + B)$$

- Values for the parameters A and B are found by minimizing the cross-entropy error function defined by a training set consisting of pairs of values $h(x^{(i)})$ and $y^{(i)}$.

Take home

- Linear vs. non-linear model.
- SVM solution = **maximum margins.**
- **Dual representation of the maximum margins problem.**
- **Kernel**
  - No need to project the input feature into the new feature space. This is saving computational cost – the "**kernel trick**".
  - Usage of selected training point for classifying new occurrences – **the support vectors** – leading to a **sparse** solution.
  - Tip: usually the RBF kernel does a good job.

Take home

- **Slack variable** to avoid over fitting ➔ **Capacity (C)** hyper-parameter that you have to set.
- **Grid Search.** Think **cross-fold validation**. Also consider Random search particularly if you have more than two hyper parameters to optimise.

AIMLab.

References

[1] Pattern Recognition and Machine Learning. Springer 2006. Christopher M. Bishop. (Chapter 7)

[2] The SVM classifier. Prof. Zisserman (Oxford). online lecture notes. http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

[3] SVM classifier applet. http://www.cns.atr.jp/~erhan/SVMclass/SVM.html

[4] LIBSVM. (a good SVM Library with MATLAB code) http://www.csie.ntu.edu.tw/~cjlin/libsvm/