

Machine Learning in Healthcare

#C10 Support Vector Machines

Technion-IIT, Haifa, Israel

Assist. Prof. Joachim Behar
Biomedical Engineering Faculty
Technion-IIT



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.

Introduction

Two class classification

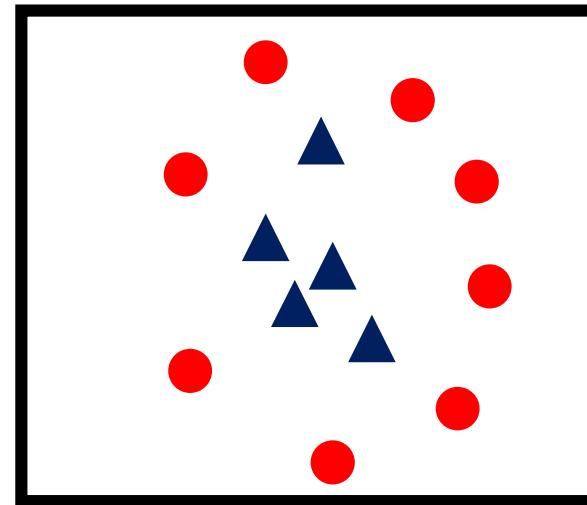
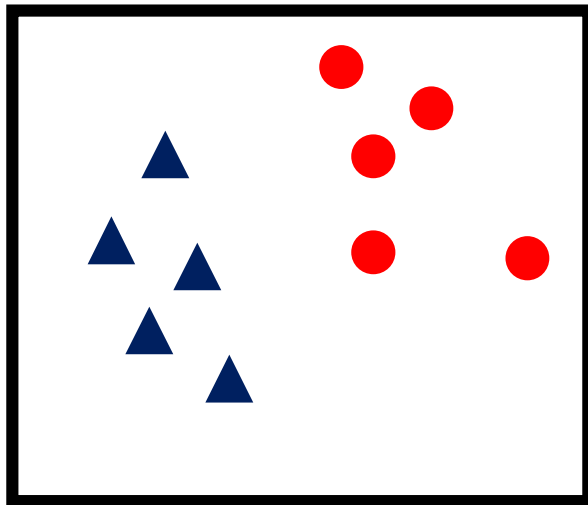
- We have a training set consisting of:
 - m examples: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
 - With target labels: $\{y^{(1)}, y^{(2)}, \dots, y^{(m)}\}$
- We wish to learn $h(x)$ so that:
 - $$h(x^{(i)}) = \begin{cases} \geq 0 & \text{for } y^{(i)} = +1 \\ < 0 & \text{for } y^{(i)} = -1 \end{cases}$$



▲ Versicolor

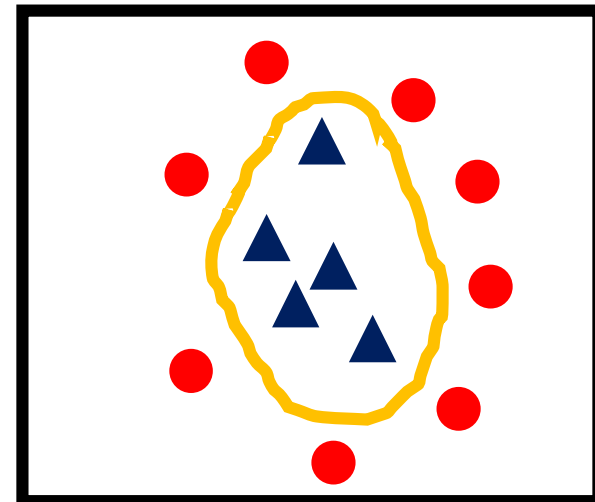
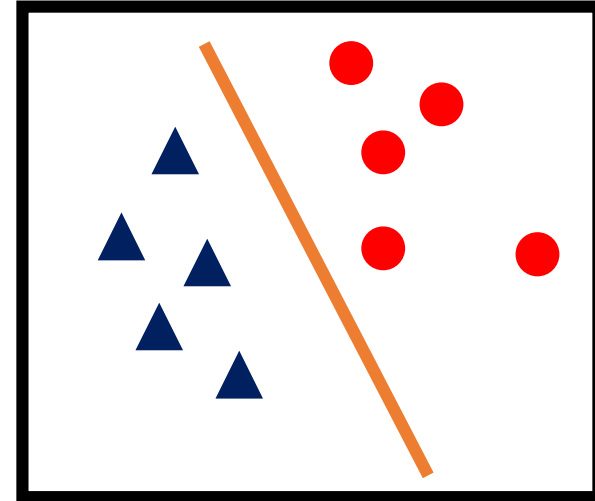


● Virginica



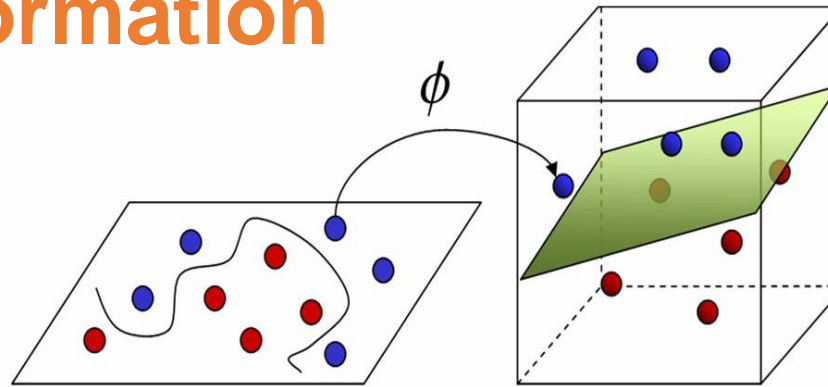
Linear versus non-linear

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



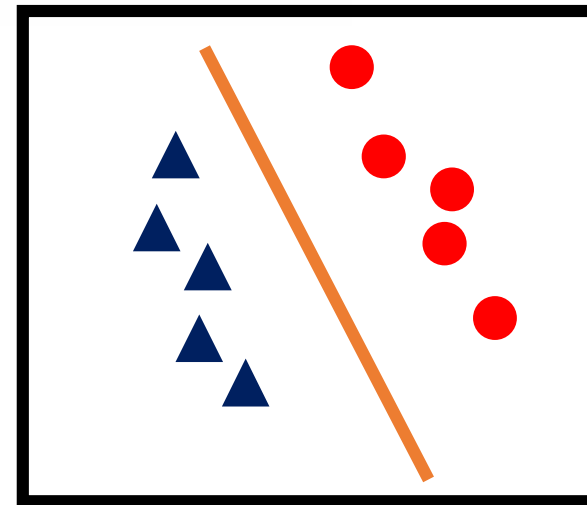
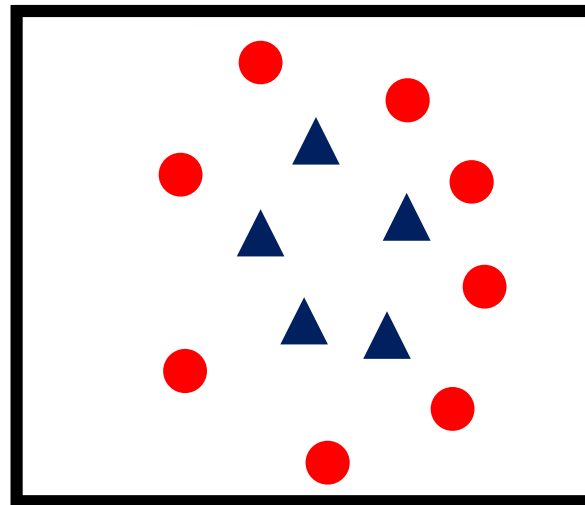
Feature space transformation

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

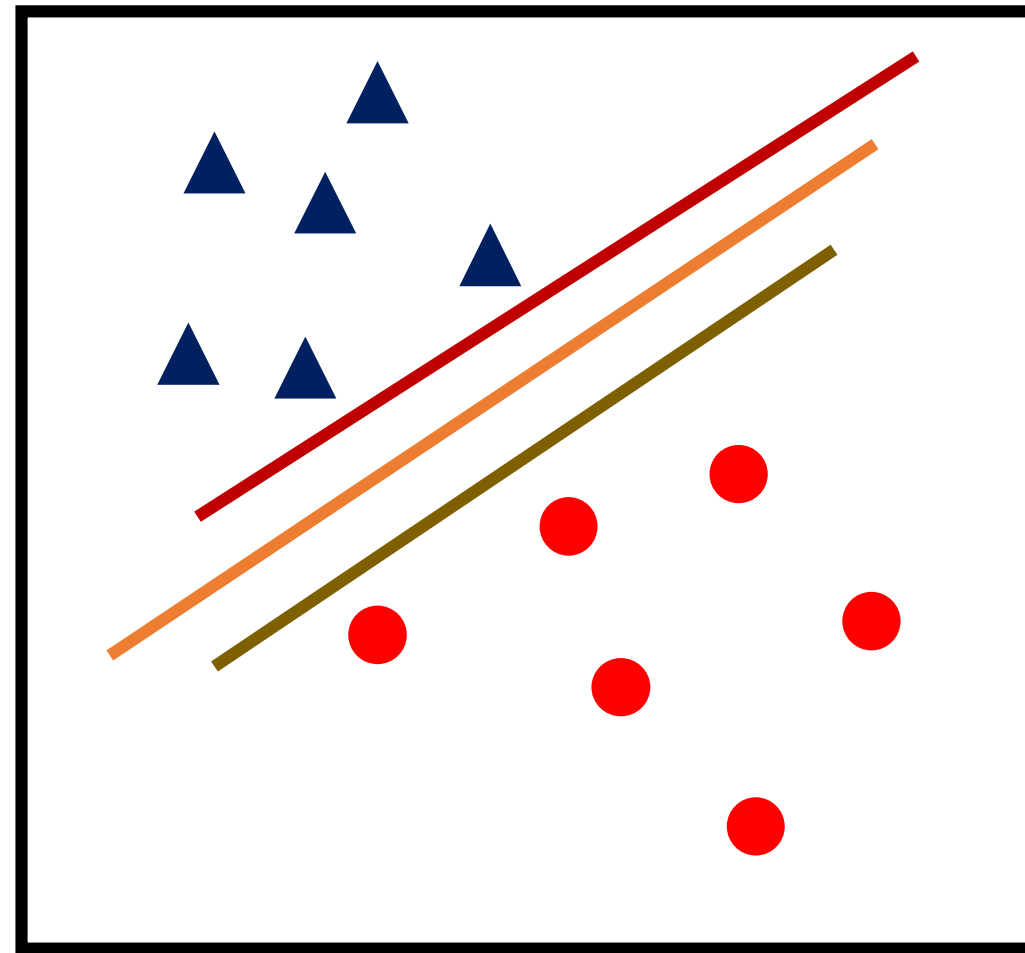
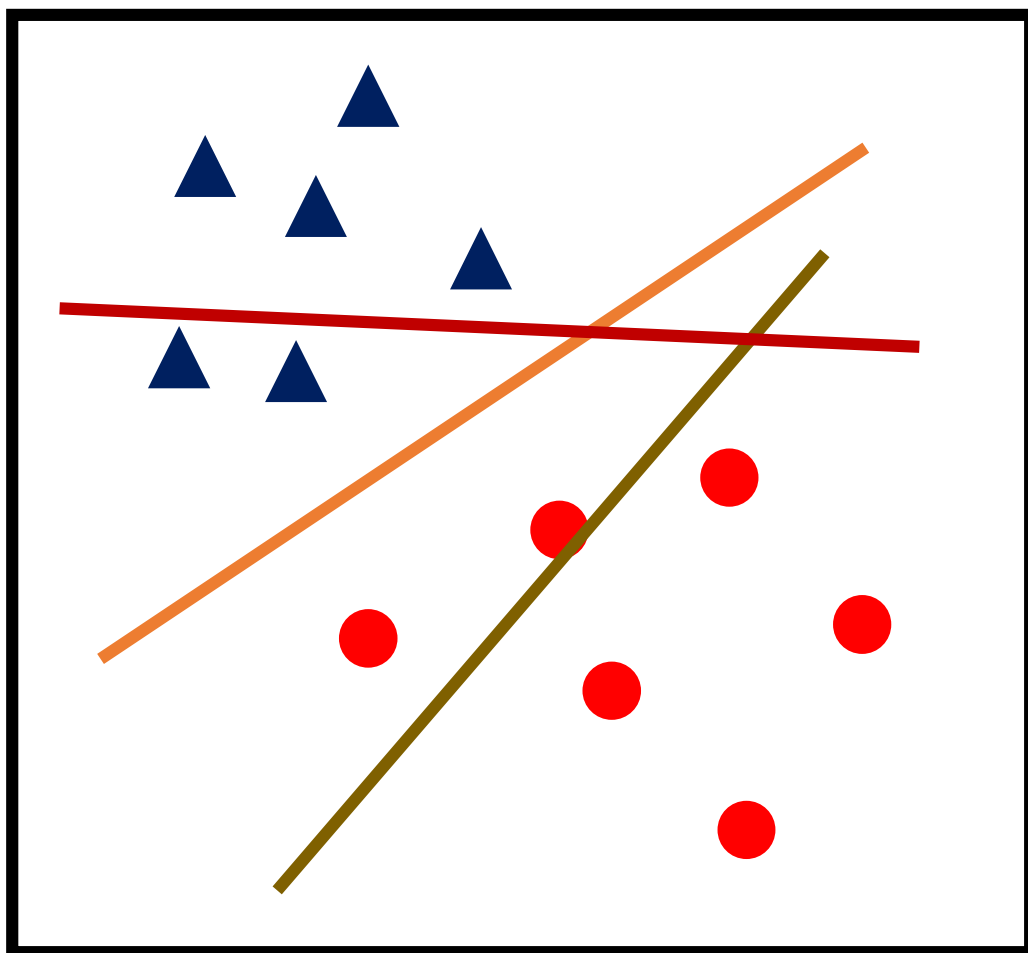


Input Space

Feature Space

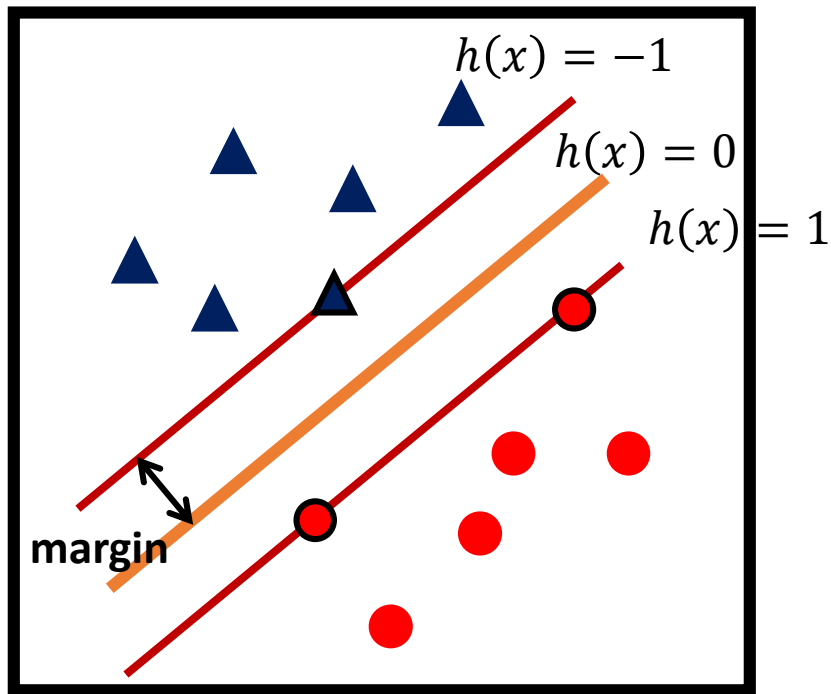


SVM - Intuition

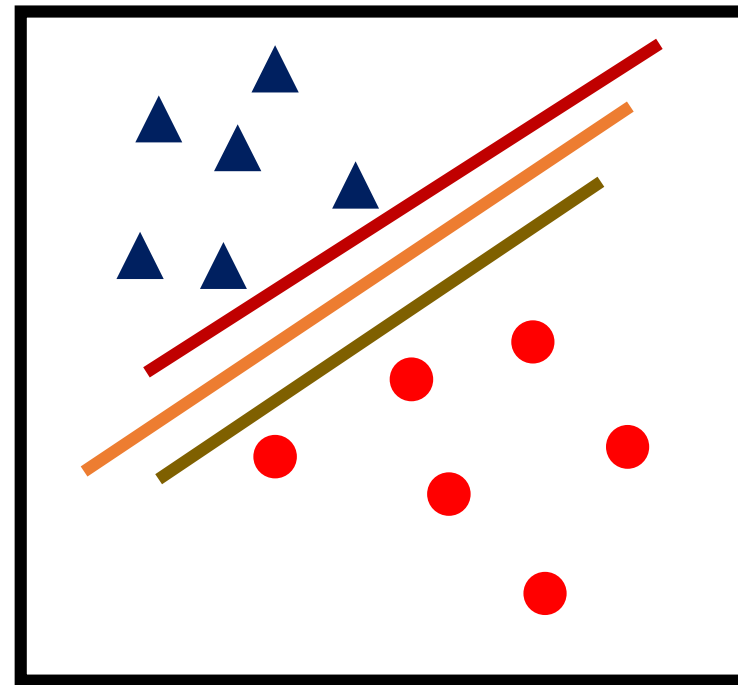


SVM – Intuition, Maximum margin

- 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

Derivation

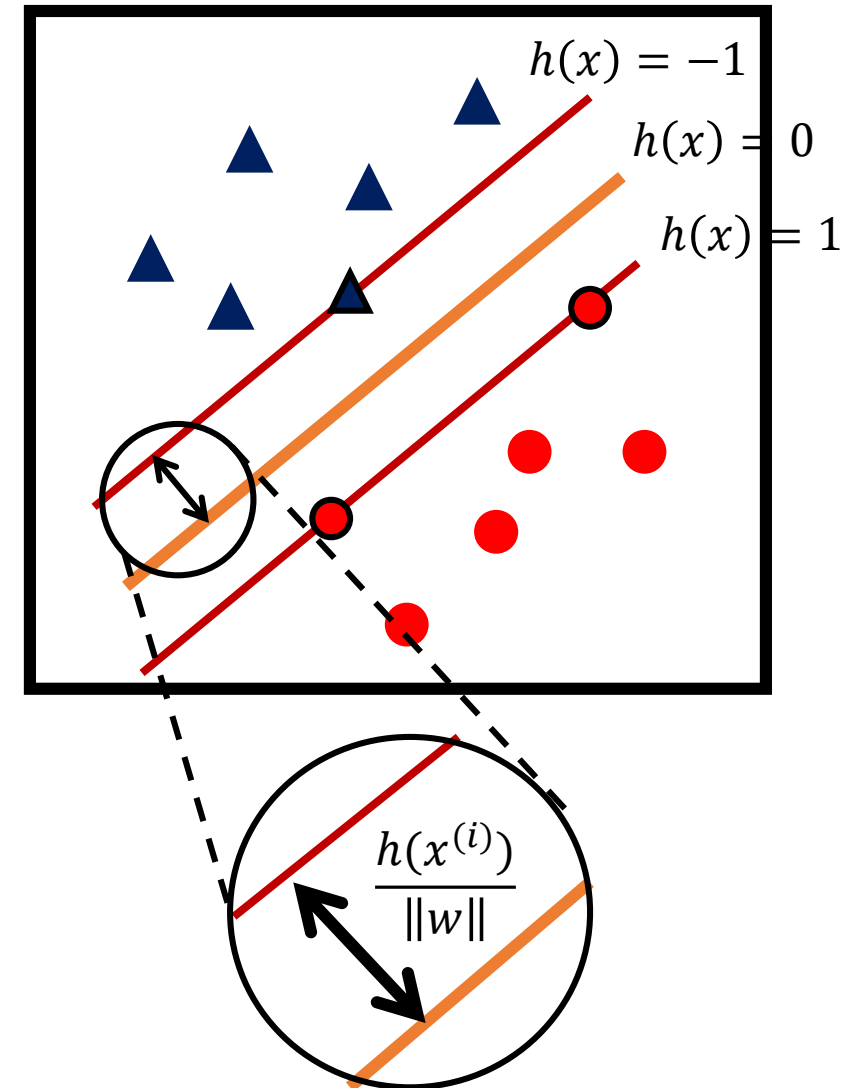
- Model: $h(x) = w^T \phi(x) + b$
- Defining margin:
 - $\frac{y^{(i)} h(x^{(i)})}{\|w\|} = \frac{y^{(i)} (w^T \phi(x^{(i)}) + b)}{\|w\|}$
- Maximum margin classifiers:

Distance to the closest point

$$\underbrace{\operatorname{argmax}_{w,b} \left\{ \frac{1}{\|w\|} \min_i [y^{(i)} (w^T \phi(x^{(i)}) + b)] \right\}}_{\text{Maximise distance}}$$

Maximise distance

$$\iff \operatorname{argmin}_{w,b} \left\{ \frac{1}{2} \|w\|^2 \right\} \\ y^{(i)} (w^T \phi(x^{(i)}) + b) \geq 1$$



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 Lagrangian function:
 - $L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m a_i \{y^{(i)}(w^T \phi(x^{(i)}) + b) - 1\}$
 - $a_i \geq 0$
- 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

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

- Under the constraints:

- $a_i \geq 0, i \in \llbracket 1..m \rrbracket$
- $\sum_{i=1}^m a_i y^{(i)} = 0$
- $w = \sum_{i=1}^m a_i y^{(i)} \phi(x^{(i)})$

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

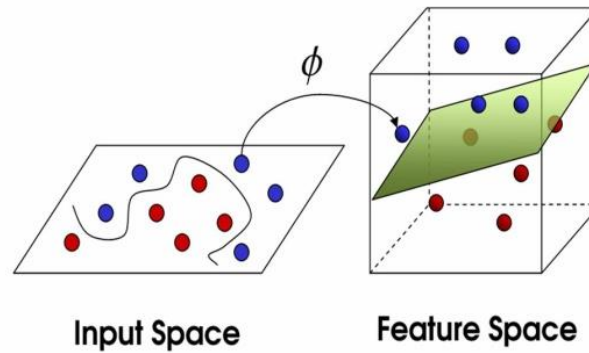


We did all that work for that. Why?

The Kernel trick

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

Remember



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

...this is computationally costly

This is where the kernel plays magic!

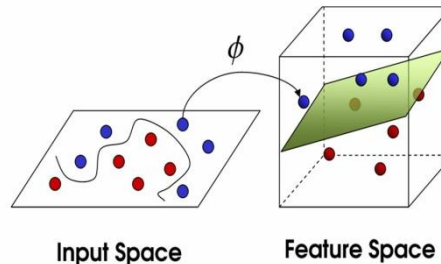
The Kernel trick

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

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.

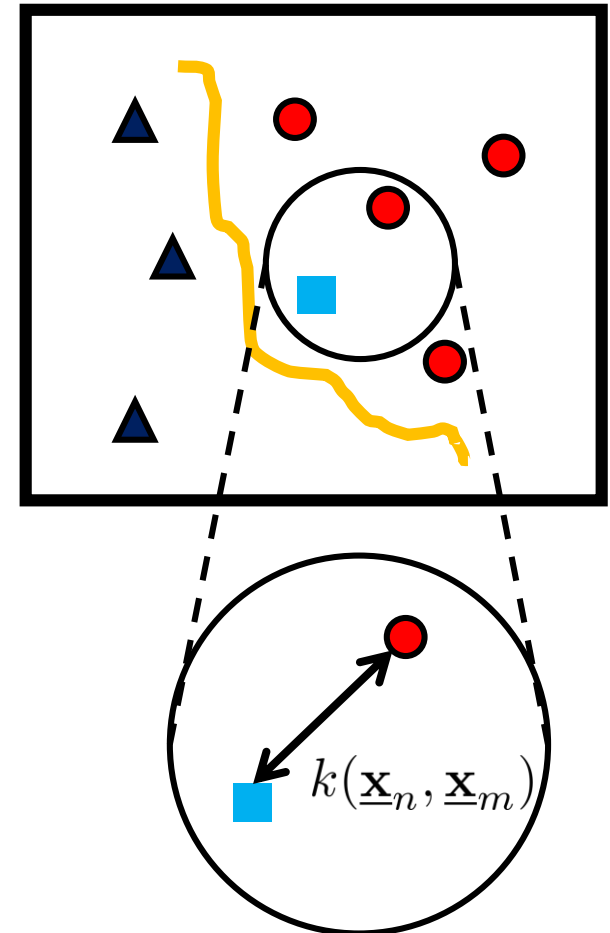
Kernel

- 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 subset of them is used during the prediction phase.

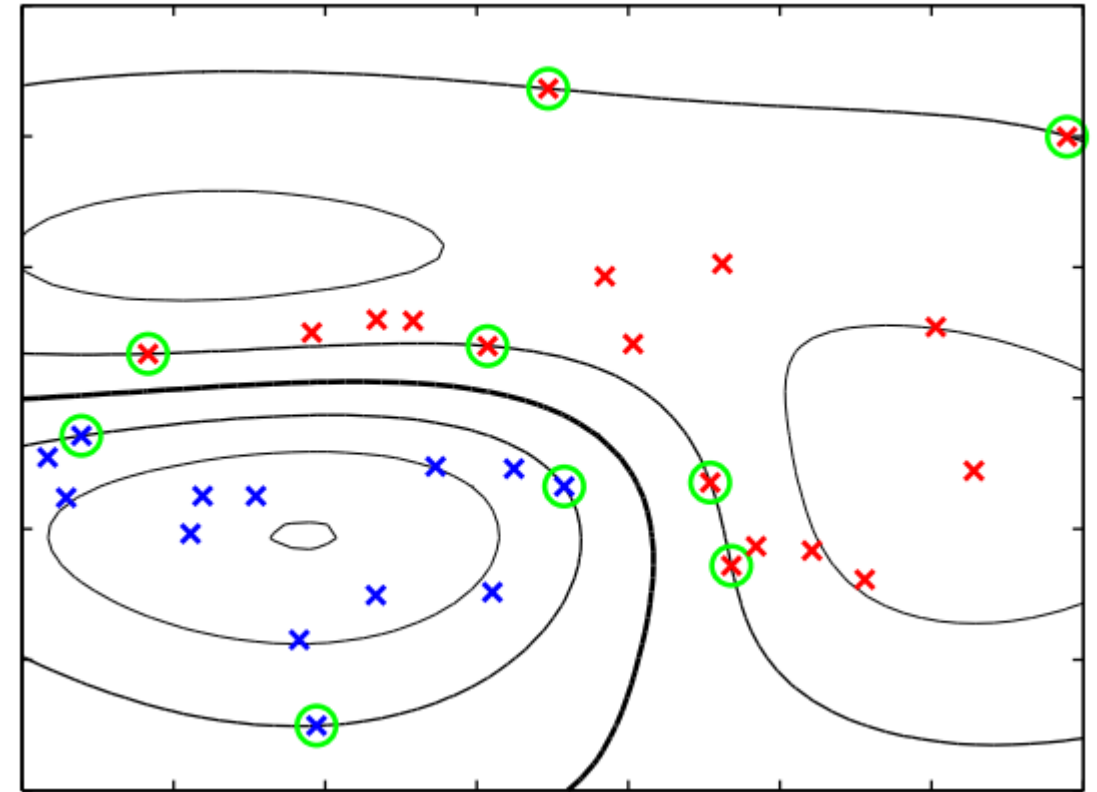


-  New occurrence
-   Support vectors



Sparsity

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

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

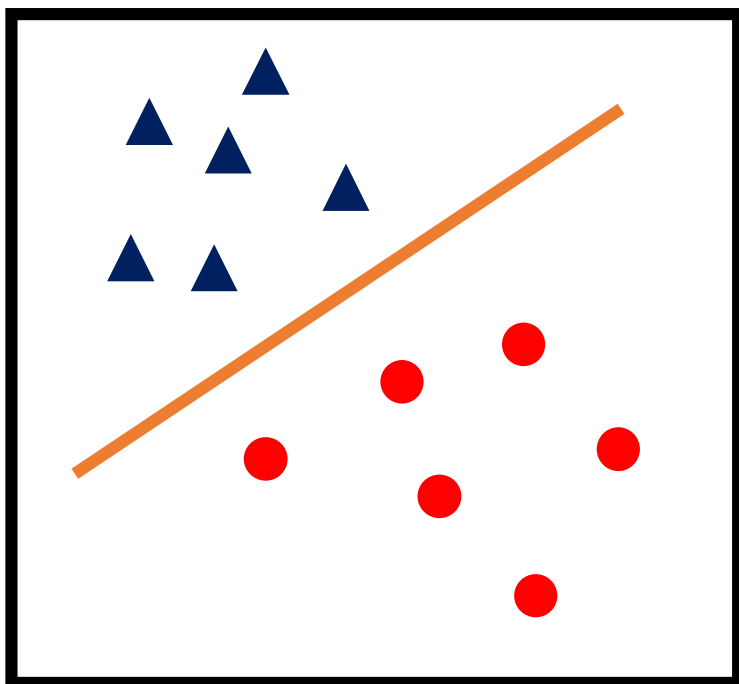
$$k(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2 / (2\sigma^2))$$

Overlapping class distribution

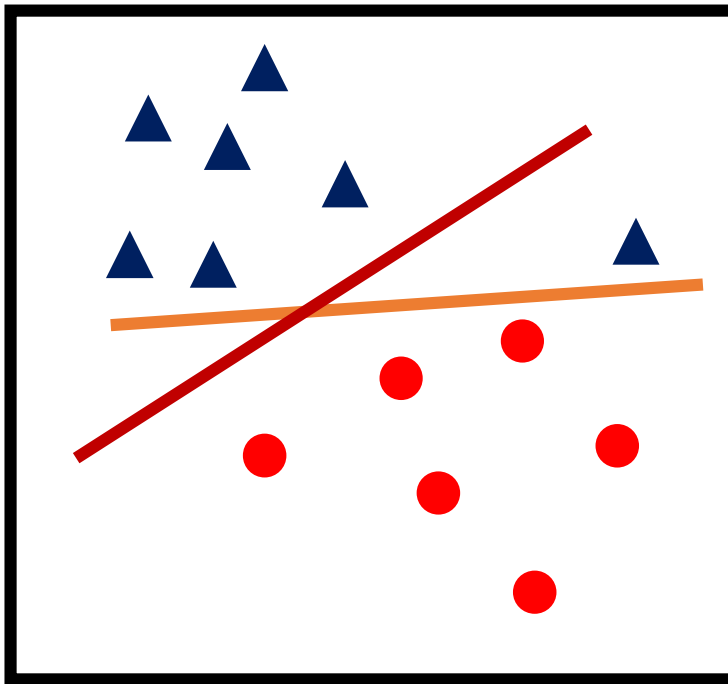
Classification

Overlapping class distribution

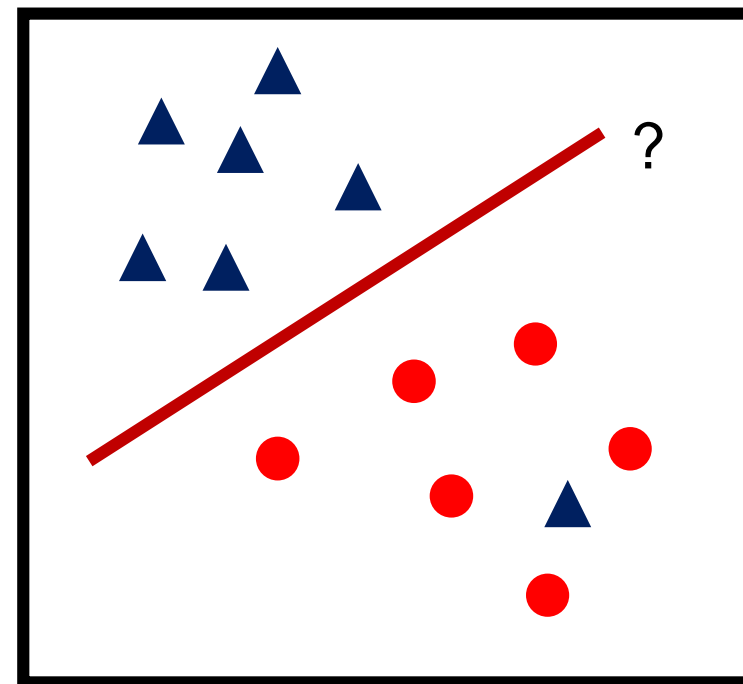
Ideal life



More real!

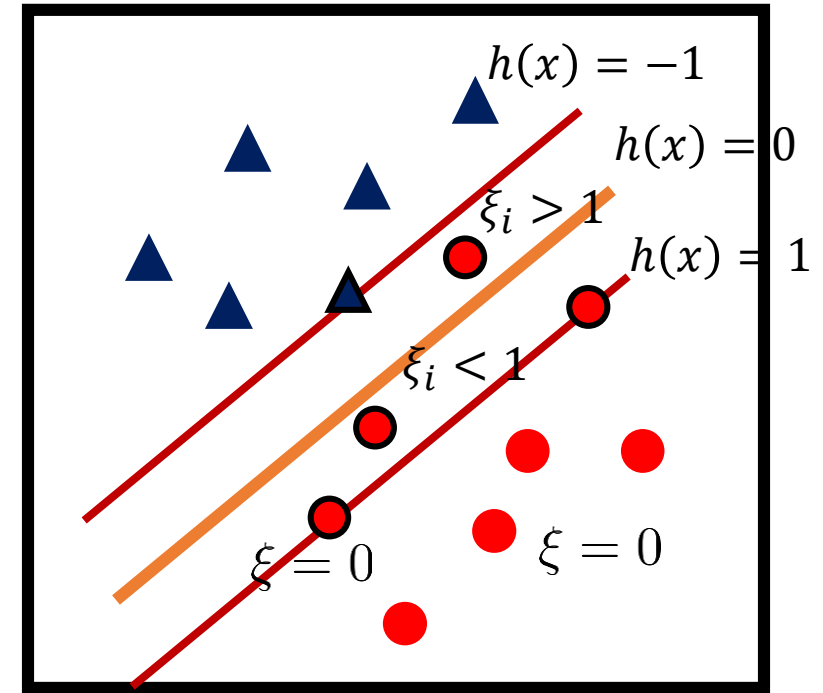


Confrontation with reality!



Overlapping class distribution

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



→ *Penalise misclassification*

→ *How much do you penalise*

Hyperparameters

Capacity and Gaussian kernel

Jupyter notebook example



- Hyperparameters:

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

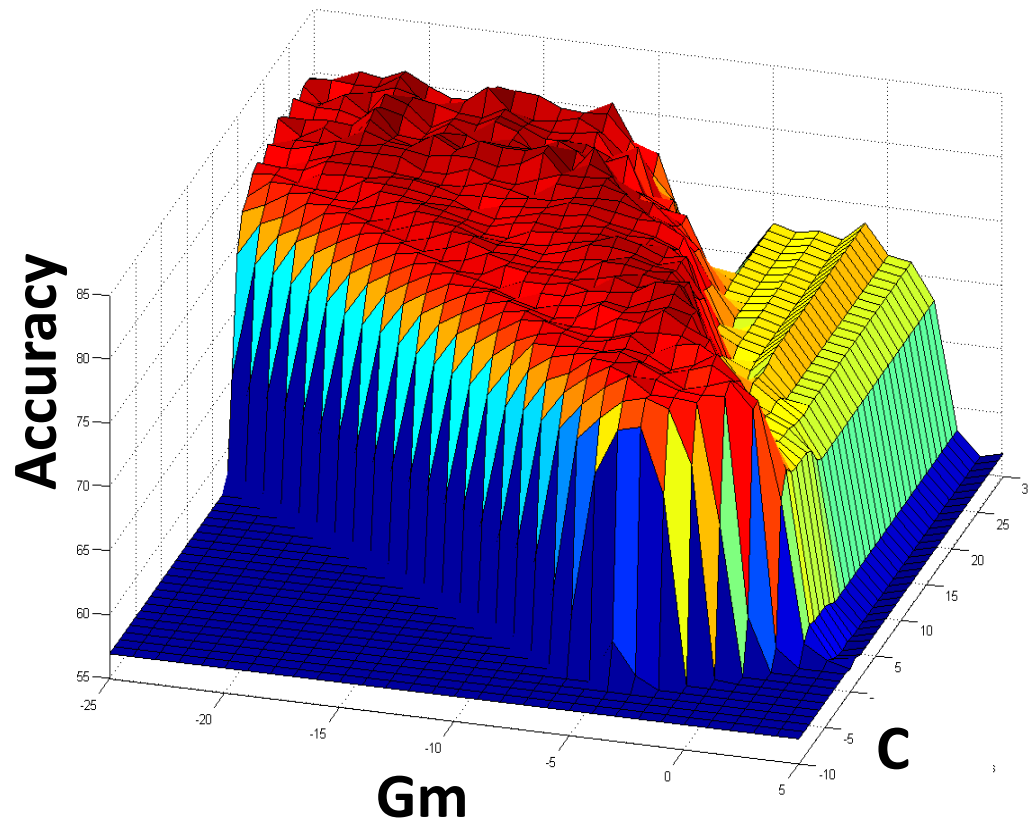
How much do you penalise

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

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



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

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

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

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.

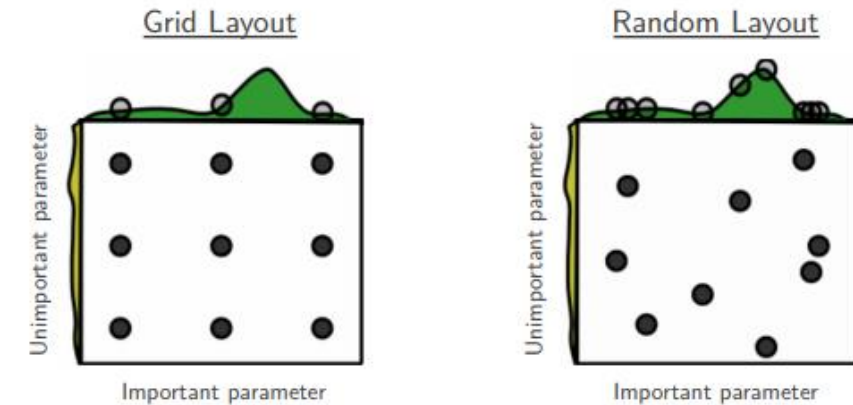
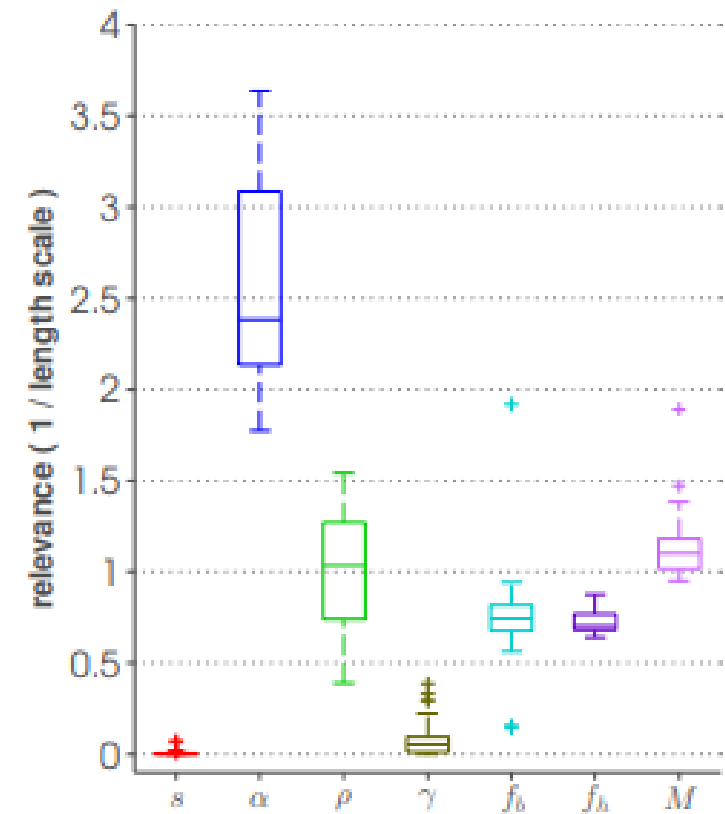
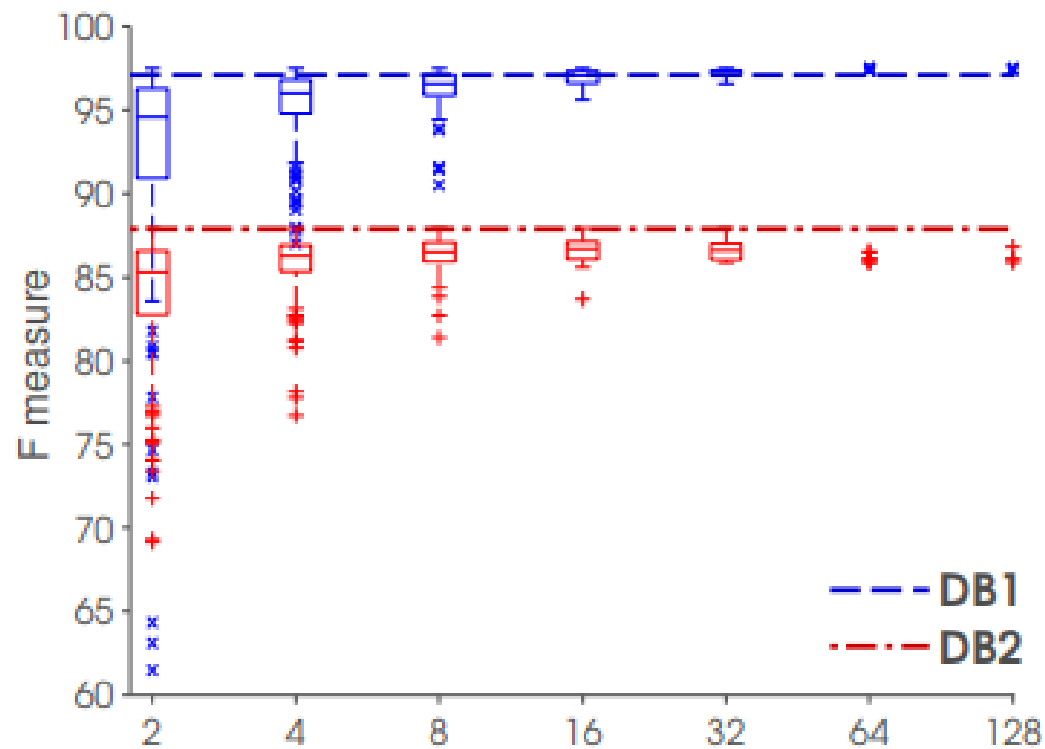


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct 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.

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

Hyper-parameters

Random search



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.
- **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.

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/>

https://www.google.com/search?q=distance+point+to+line&rlz=1C1GCEU_enIL849IL849&sxsrf=ACYBGNRc-FsqiK0DE9F21eA1FEsqgOf5kQ:1575194896010&source=lnms&tbn=isch&sa=X&ved=2ahUKEwiZ4emhmpTmAhXKa1AKHcpADy0Q_AUoAXoECAwQAw&biw=1463&bih=713#imgsrc=FkHnGUcvEH2nCM:

