Categorisation by
Kernel Machine/Support Vector Machine

Tae-Kyun Kim
Senior Lecturer
http://www.iis.ee.ic.ac.uk/ComputerVision/

Backgrounds:
Vector calculus - Matrix and vector derivatives
Optimisation (EE429) - Gradient method

Notations

x : feature vector for classification e.g. the *K* binned bag-of-words histogram

K : dimension of input vector **x**

N : number of training data vectors **x**

I : image

M : number of classes

$\boldsymbol{c}_m$: m-th class

y(**x**) : SVM output (before discretization)

ϕ(**x**) : nonlinear (kernel) mapping

K': dimension of feature space after kernel mapping

$k(\mathbf{x}_1, \mathbf{x}_2)$: kernel function

t_n : binary target variable of $\boldsymbol{x}_n$

N_{sv} : number of support vectors

ξ_n *: slack variables*

C : trade-off constant for misclassified samples

Categorization by SVM

- Once descriptors have been assigned to form feature vectors, we reduce the problem of generic visual categorization to that of multi-class classification, with as many classes as defined visual categories.
- The categorizer performs two separate steps in order to predict the classes of images: training and testing.
- During training, (class) labeled data is sent to the classifier and used to learn a statistical decision procedure for distinguishing categories.
- Among many available classifiers, including NN (Nearest Neighbor) classifier, we adopt the SVM (Support Vector Machine).
- It is often known to produce state-of-the-art results in high-dimensional problems.

Categorization by SVM

- The SVM classifier finds a hyperplane (or decision boundary) which separates two-class data.

- For given observations **x**, and corresponding labels t which takes values ±1, it finds a classification function:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

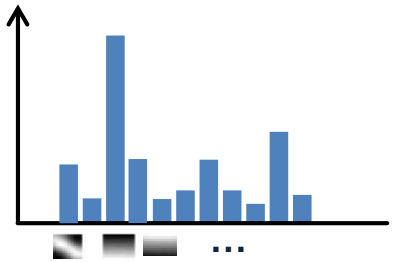
where **w**, *b* represent the parameters of the hyperplane.

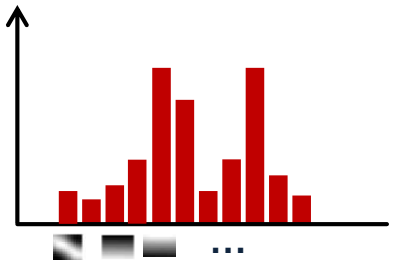
- Data points **x** are classified by the sign of y(**x**).

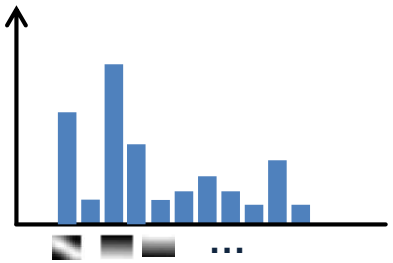
- In an example case with bag of words, the input feature vector **x** is the *K* binned histogram formed by the number of occurrences of each codeword in the image **I**.

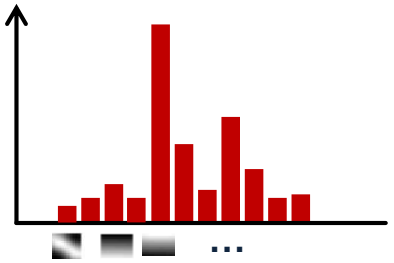
- In order to apply SVM to multi-class problems, we take the multi-class extensions of SVM.

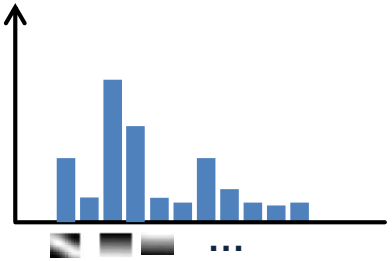
Category model

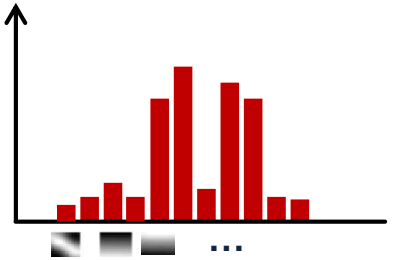








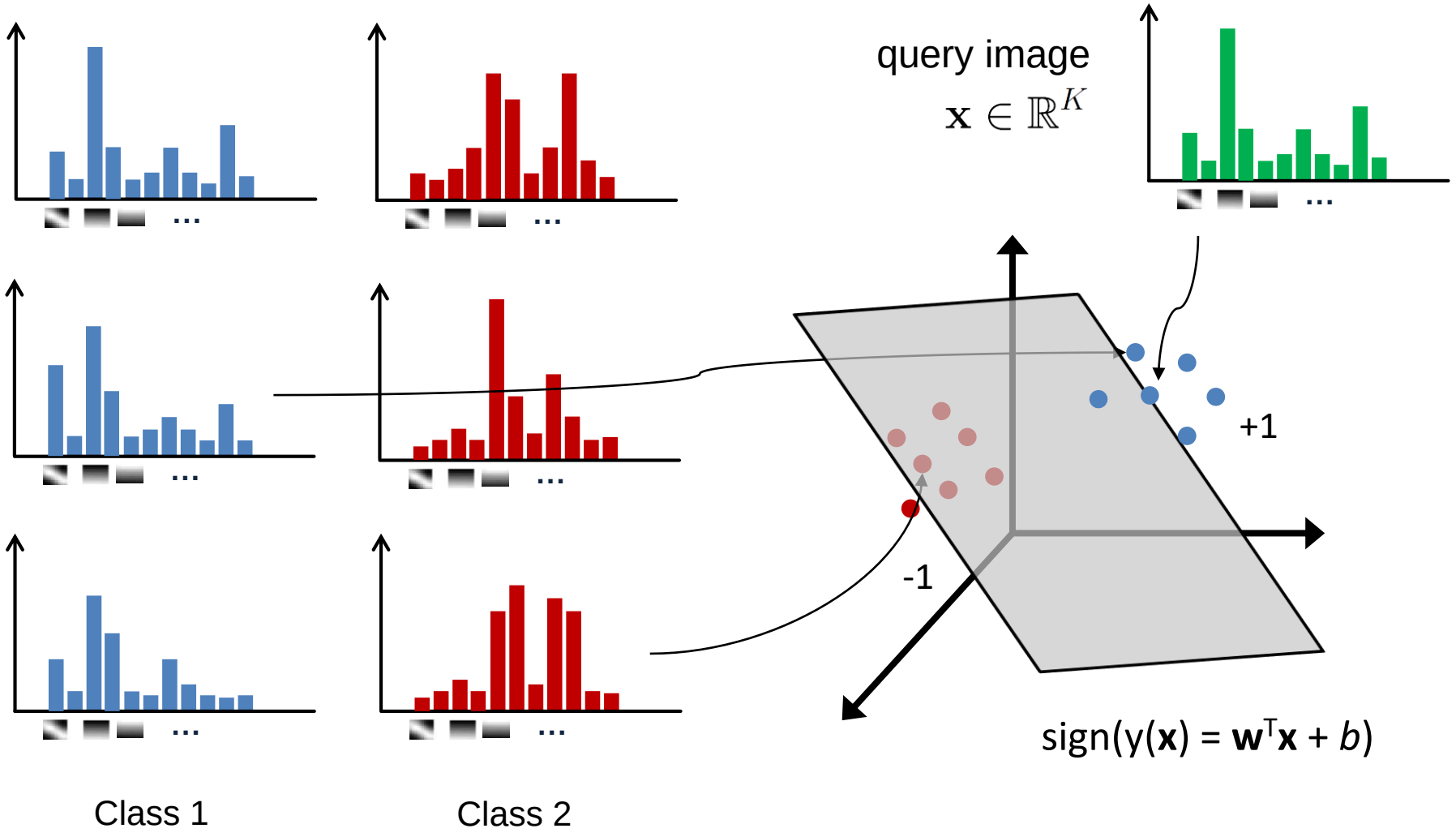




I x I x

Class 1 Class 2

Discriminative classifier (linear classifier)



Support Vector Machine
(Maximum Margin Classifier or Sparse Kernel Machine)

Backgrounds:

Vector calculus - Matrix and vector derivatives

Optimisation (EE429) - Gradient method, Lagrange multipliers (KKT conditions)

Further reading:

Appendix A: Mathematical Foundations, R.Duda, P.Hart, D.Stork, Pattern Classifcation (Second Edition), JOHN WILEY & SONS, Inc. 2001.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.320.4607&rep=rep1&type=pdf

http://cns-classes.bu.edu/cn550/Readings/duda-etal-00.pdf

Chapter 7, C.M.Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

Christopher J.C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition (http://research.microsoft.com/pubs/67119/svmtutorial.pdf)

Linear Discriminant Functions

- A discriminant function maps an input vector **x** into one of *M* classes, denoted $\mathcal{C}_m$.

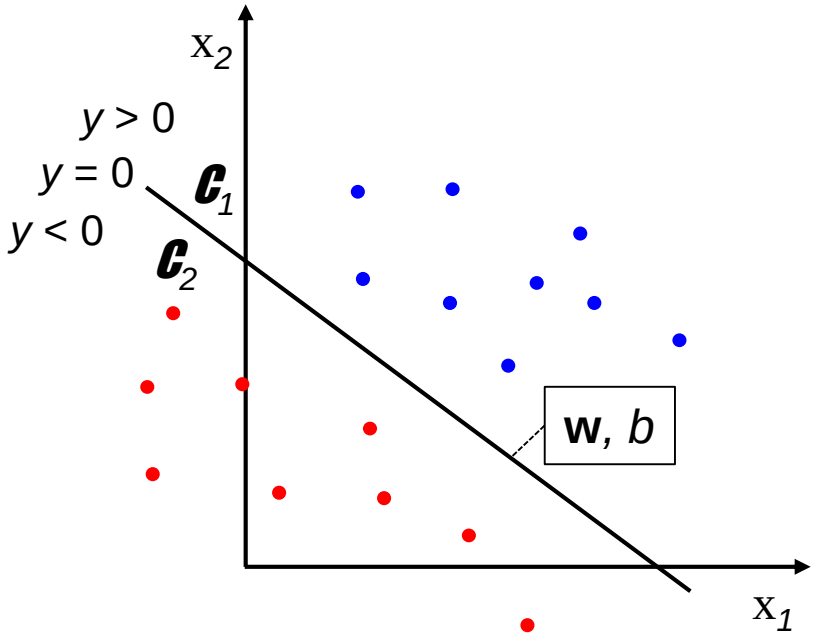
  For simplicity, consider two classes.

- *Linear* discriminant function takes the form of

  $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$

  where **w** is a *weight vector* and *b* a *bias*.

- A vector **x** is assigned to $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$, and $\mathcal{C}_2$ otherwise.



Linear Discriminant Functions

- The decision boundary is defined by $y(\mathbf{x}) = 0$, which is a $(K\text{-}1)$-dimensional hyperplane in the K-dimensional input space.
- Consider $\mathbf{x}_A$ and $\mathbf{x}_B$ on the decision surface:

$$\rightarrow \quad y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0$$

$$\rightarrow \quad \mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$$

- The vector $\mathbf{w}$ is orthogonal to the decision surface. $\mathbf{w}$ determines the direction of the decision surface.
- For $\mathbf{x}$ on the decision surface, $y(\mathbf{x}) = 0$, so the normal distance from the origin to the decision surface is

$$\frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||} = \frac{-b}{||\mathbf{w}||}$$

- Thus, b determines the location of the decision surface.

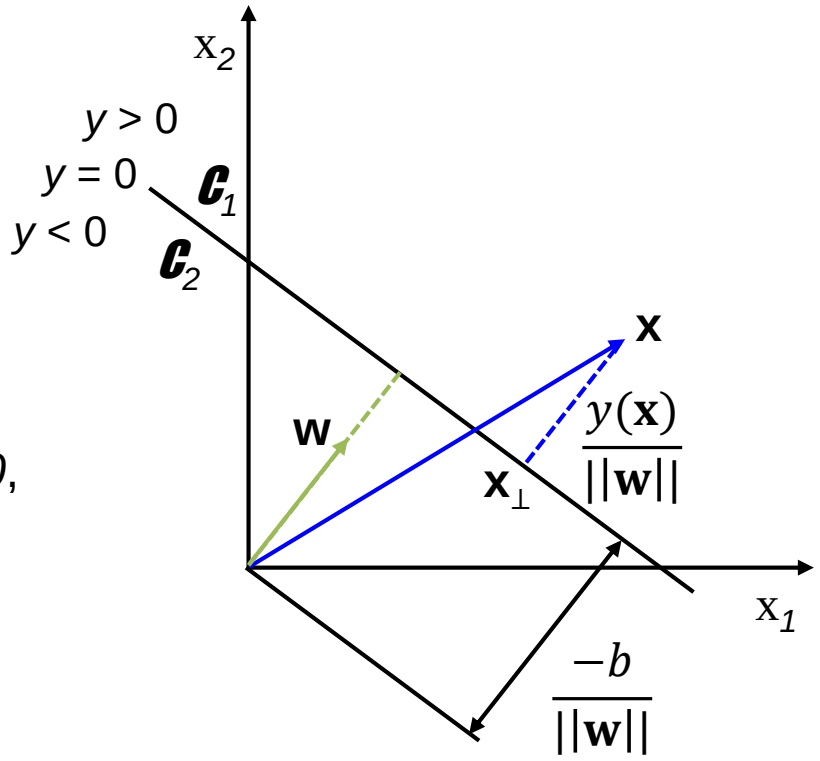
Linear Discriminant Functions

- For an arbitrary point **x**, its orthogonal projection onto the decision surface $\mathbf{x}_\perp$ is such that

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{||\mathbf{w}||}.$$

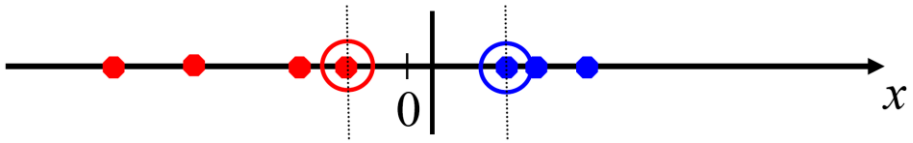
- Multiply both sides by $\mathbf{w}^T$ and add *b*, and use $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ and $y(\mathbf{x}_\perp) = \mathbf{w}^T\mathbf{x}_\perp + b = 0$, we have

$$r = \frac{y(\mathbf{x})}{||\mathbf{w}||}.$$

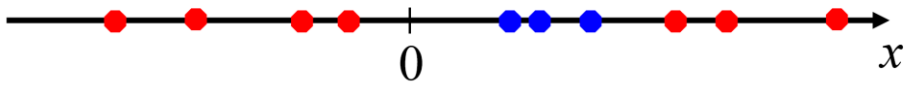


Kernel Trick

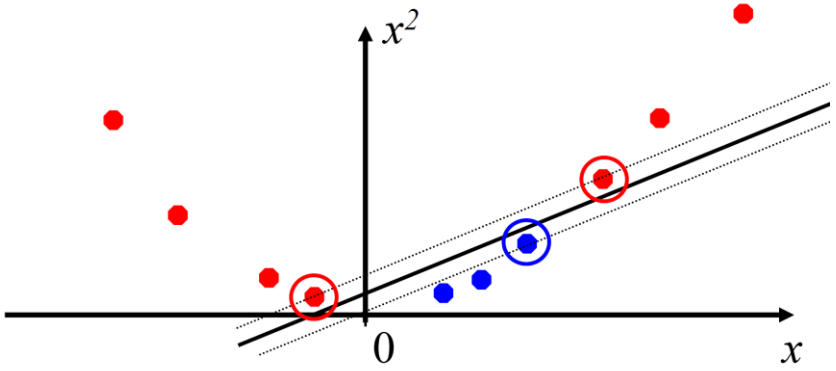
– Linear discriminant functions separate data linearly:



– Dataset is often not linearly separable:

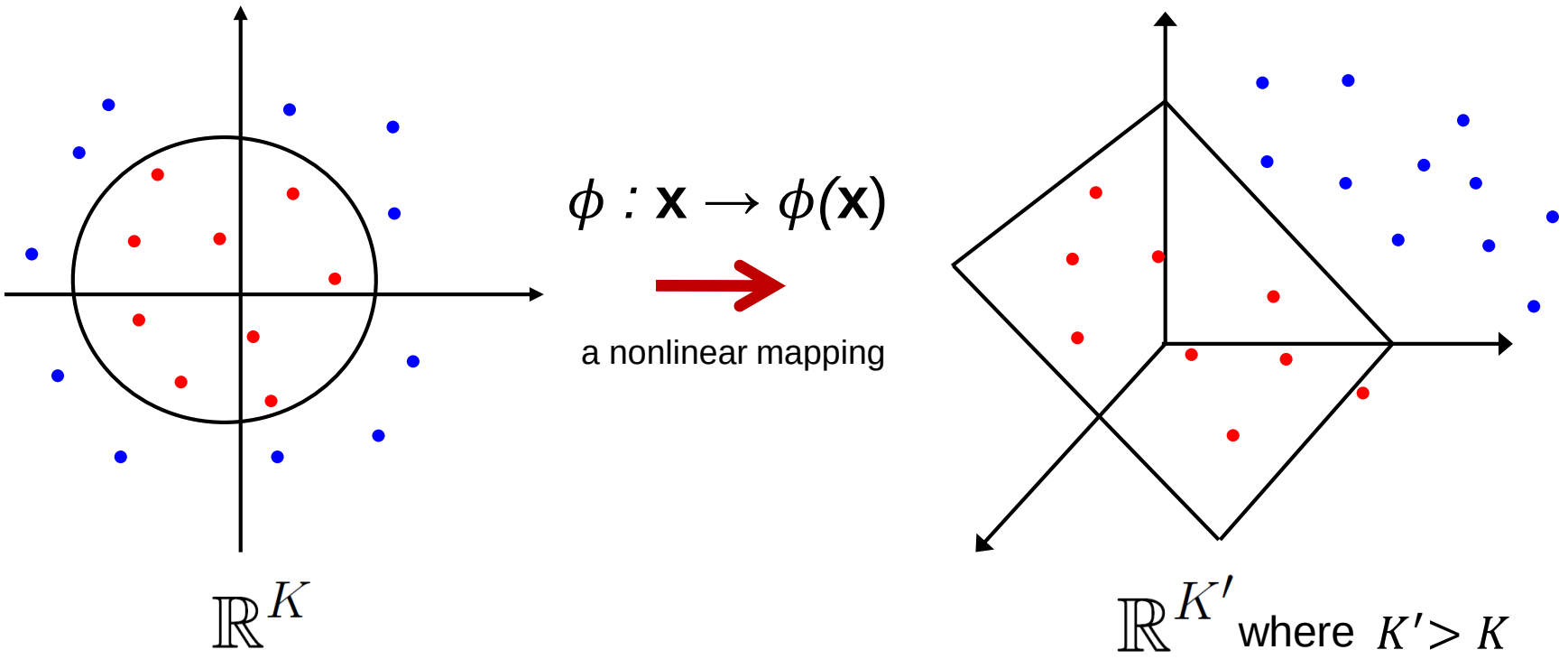


– We can map it to a higher-dimensional space, then separate it linearly:



Kernel Trick

- The input space is transformed into a higher-dimensional feature space by a nonlinear mapping $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x})$, where a linear decision boundary can separate all data points.
- This second feature space may have a high or even infinite dimension.



Kernel Trick

– One of the advantages of SVM is that it can be formulated entirely in terms of scalar products in the second feature space, by introducing the kernel function

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

– The kernel is a symmetric function s.t. $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$

– NOTE:
  • It is not required to compute $\phi(\mathbf{x})$ explicitly.

– Examples of the kernel functions are:
  • Linear kernel: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$
  • Polynomial kernel: $k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T \mathbf{x}' + c\right)^M$ with $c > 0$
  • Gaussian (or RBF) kernel: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$

Support Vector Machine

- The SVM classifier finds a hyperplane which separates two-class data with maximal margin.
- The margin is defined as the distance of the closest training point to the separating or decision hyperplane.
- Thus it is also called maximum margin classifier.

- It can be shown that the support vectors are those feature vectors lying nearest to the decision hyperplane.

- SVM has sparse solutions i.e. the predictions for new inputs depend only on the kernel function evaluated at a subset of the training data points, which are called support vectors.
- Thus it is also called sparse kernel machine.

Support Vector Machine

– We have N training data vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and their target values $t_1, \ldots, t_N$ where $t_n \in \{-1, 1\}$.

– SVM for the two-class problem takes the form of

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

where $\phi(\mathbf{x})$ denotes the feature mapping, $\mathbf{w}$ is the weight vector and b the bias.

– Data points $\mathbf{x}$ are classified by the sign of $y(\mathbf{x})$.

– Assume that the training data is linearly separable in the feature space. Thus, optimal $\mathbf{w}$ and b satisfy

$$t_n y(\mathbf{x}_n) > 0$$

for all training data points.

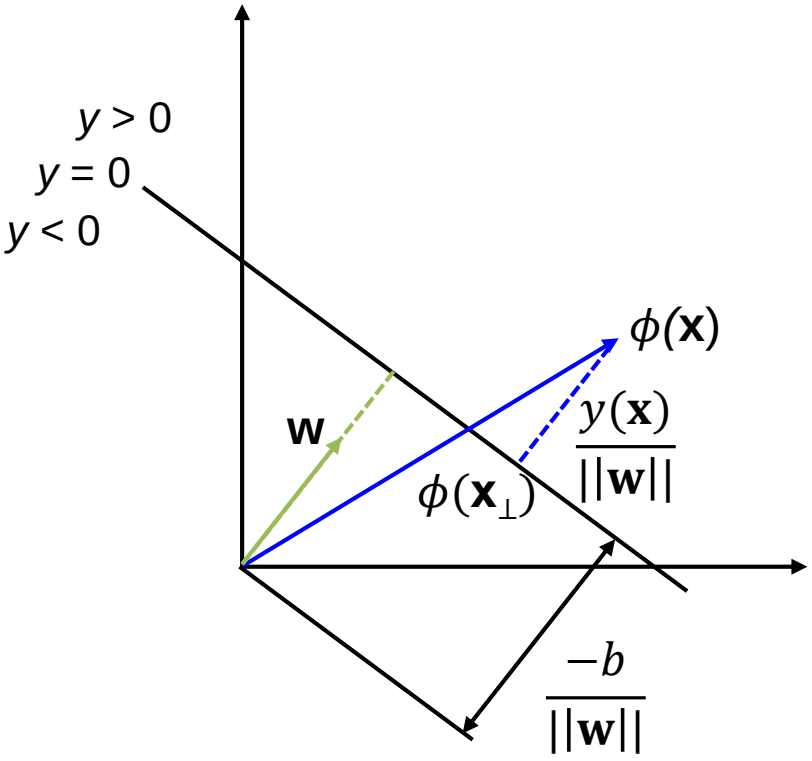
Support Vector Machine

- The perpendicular distance of a point **x** from a hyperplane $y(\mathbf{x})$ is

$$|y(\mathbf{x})|/||\mathbf{w}||$$

- As we assume $t_n y(\mathbf{x}_n) > 0$ for all n, the distance of a point $\mathbf{x}_n$ to the decision surface is

$$\frac{t_n y(\mathbf{x}_n)}{||\mathbf{w}||} = \frac{t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b)}{||\mathbf{w}||}$$



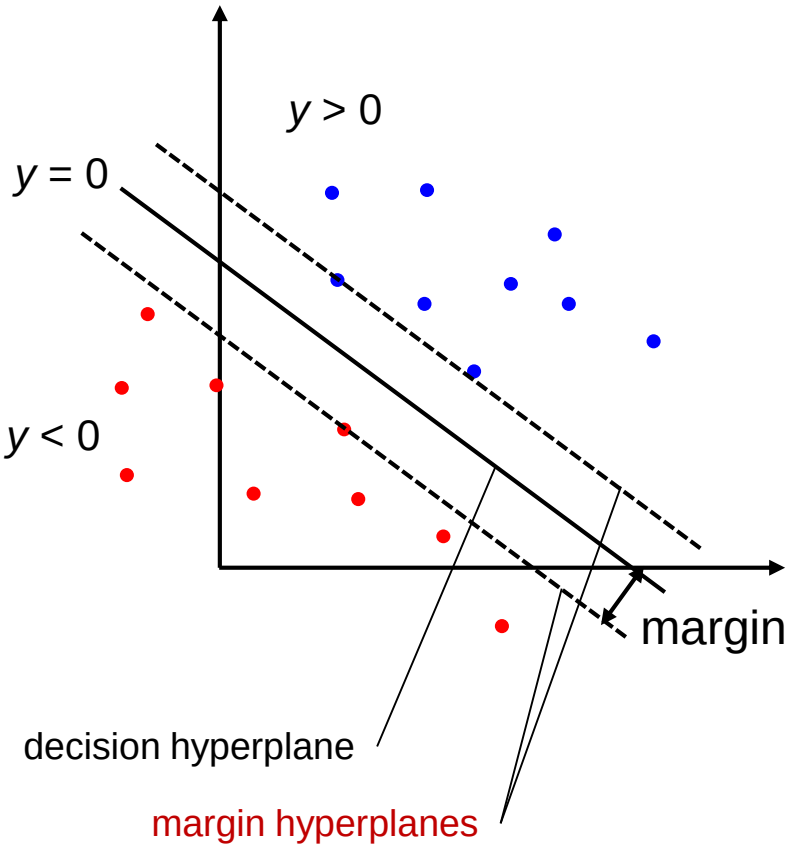
Support Vector Machine

- The margin is the minimum perpendicular distance:

$$\frac{1}{||\mathbf{w}||} \min_n [t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b)]$$

- We find **w** and *b* that maximise the margin i.e.

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||} \min_n [t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\}$$



Canonical representation of decision hyperplane

- Rescaling **w** → s**w** and b → sb does not change the distance from any point $\mathbf{x}_n$ to the decision hyperplane.

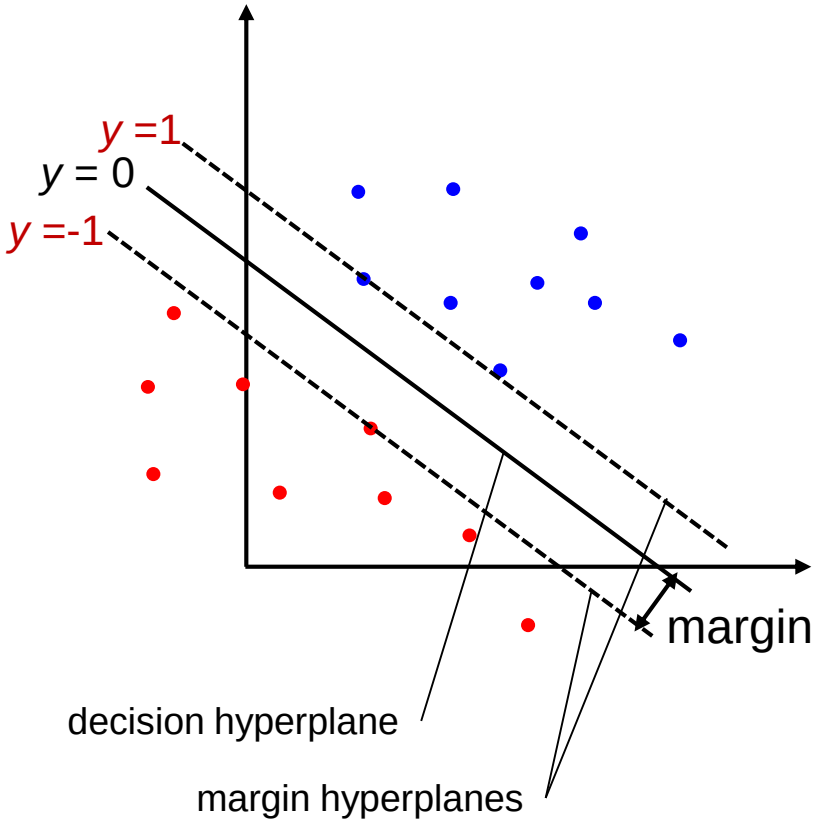
- We can therefore set

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) = 1$$

for the point that is closest to the hyperplane.

- All data points satisfy

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, ..., N$$



Optimisation

– The original problem $\arg\max_{\mathbf{w},b}\left\{\frac{1}{||\mathbf{w}||}\min_n[t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)]\right\}$

becomes to maximise $||w||^{-1}$.

– Equivalently, we have $\arg\min_{\mathbf{w},b}\frac{1}{2}||\mathbf{w}||^2$

subject to the constraints $t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)\geq 1, \quad n=1,...,N$

– The problem is solved by Lagrange multipliers $a_n \geq 0$:

$$L(\mathbf{w},b,\mathbf{a})=\frac{1}{2}||\mathbf{w}||^2-\sum_{n=1}^{N}a_n\{t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)-1\}$$

– Note the minus sign in front of the Lagrange multiplier term, as we are minimising the function.

Optimisation

– Setting the derivative of $L(\mathbf{w}, b, a)$ w.r.t. $\mathbf{w}$ to zero, we obtain

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

– The SVM decision function $y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$

takes the form of $y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$

where $a_n \geq 0$, $\mathbf{x}_n$ are the training data, t_n are the label, b is the bias, and

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

Optimisation

- In the above,
  - a_n are typically zero for most *n*.
  - Equivalently, the sum can be taken only over a selected few of $\mathbf{x}_n$.
  - These vectors are known as support vectors.
  - It can be shown that the support vectors are those vectors lying on the margin hyperplanes.

- Computational complexity for classifying a new data point **x** (linear vs nonlinear SVM)
  - Whereas the nonlinear SVM requires O(KxN$_{sv}$), the linear SVM takes O(K)

  where K is the data vector dimension and N$_{sv}$ is the number of support vectors.

Nonlinear:

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b \qquad \text{where } k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$$

Linear:

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n \mathbf{x}^T \mathbf{x}_n + b = \mathbf{x}^T \sum_{n=1}^{N} a_n t_n \mathbf{x}_n + b$$

pre-computed

Optimisation

– Karush-Kuhn-Tucker (KKT) conditions of the generic Lagrangian function are

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}).$$

KKT conditions →

$$\begin{aligned} \lambda &\geq 0, \\ g(\mathbf{x}) &\geq 0, \\ \lambda g(\mathbf{x}) &= 0. \end{aligned}$$

– The KKT conditions for SVM are

$$a_n \geq 0$$

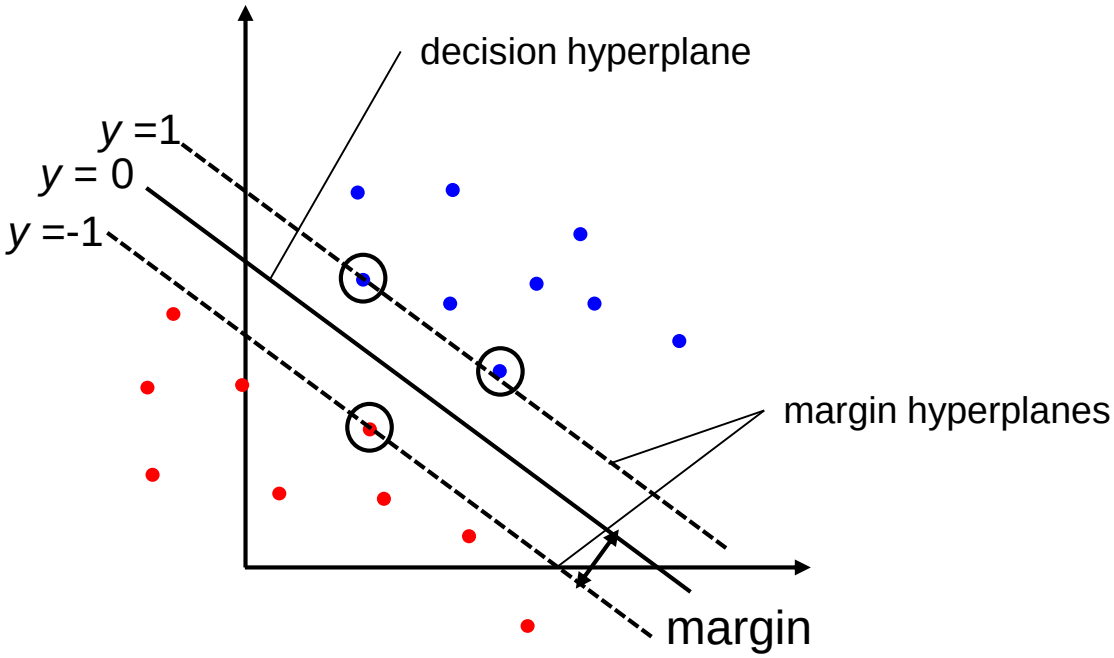
$$t_n y(\mathbf{x}_n) - 1 \geq 0$$

$$a_n \{t_n y(\mathbf{x}_n) - 1\} = 0.$$

→

- For every data point, a_n=0 or $t_n y(\mathbf{x}_n)$ =1.
- Any data point for which a_n=0 plays no role in making predictions.
- The remaining data points are called support vectors, and they satisfy $t_n y(\mathbf{x}_n)$ =1, i.e. lying on the margin hyperplanes.

Optimisation



Support vectors (in circle) lie on the margin hyperplanes.

Overlapping class distributions

- Data sets are often not linearly separable. The SVM takes two approaches to this problem.
- Firstly a mapping $\phi(\mathbf{x})$ is made from the original data space of $\mathbf{x}$ to a higher-dimensional feature space.
- Secondly it introduces an error weighting constant C which penalizes misclassification of samples in proportion to their distance from the classification boundary.

- Slack variables, $\xi_n \geq 0$, n=1,...,N are defined s.t.

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, \quad n = 1, \ldots, N.$$

$\xi_n = 0$ for data points that are on or inside the correct margin boundary and $\xi_n = |t_n - y(\mathbf{x}_n)|$ for other points.

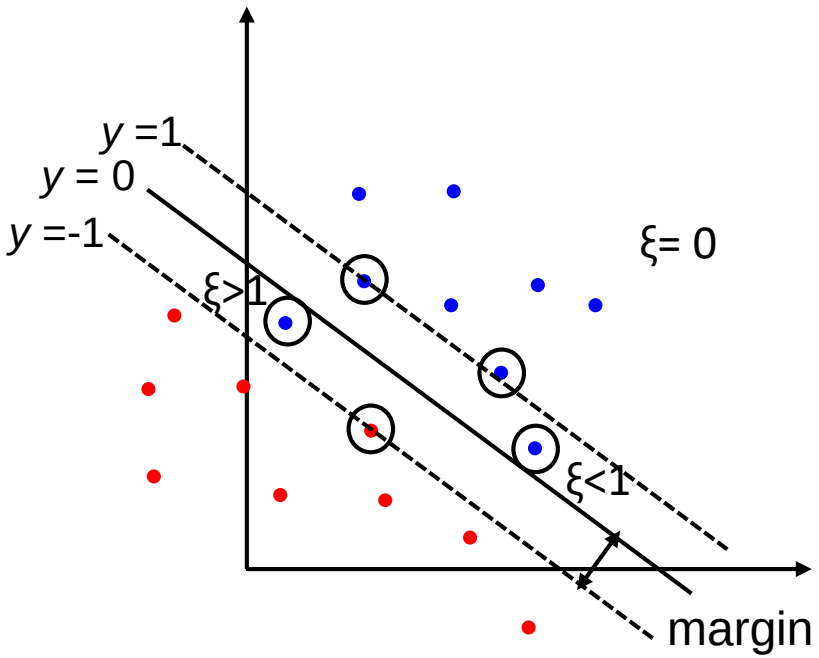
Overlapping class distributions

– We therefore minimise

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}||\mathbf{w}||^2$$

where $C > 0$ is a trade-off constant and $\xi_n \geq 0$, subject to

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, \quad n = 1, ..., N.$$

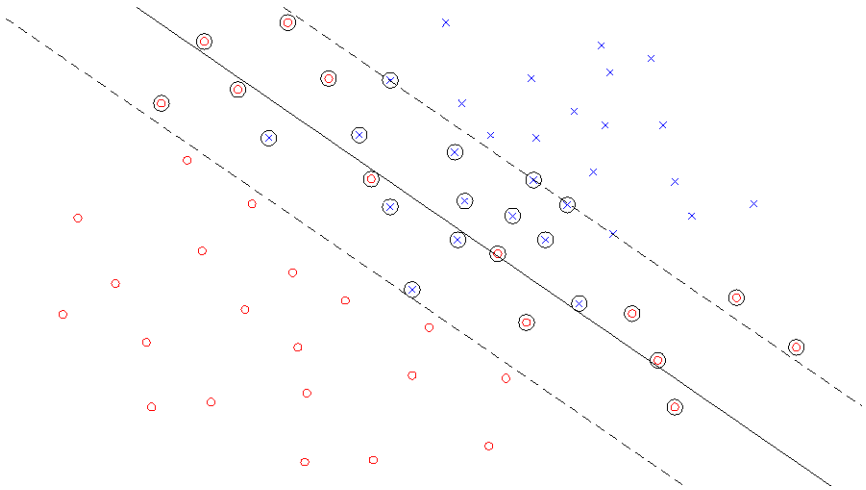


Support vectors are shown in circle

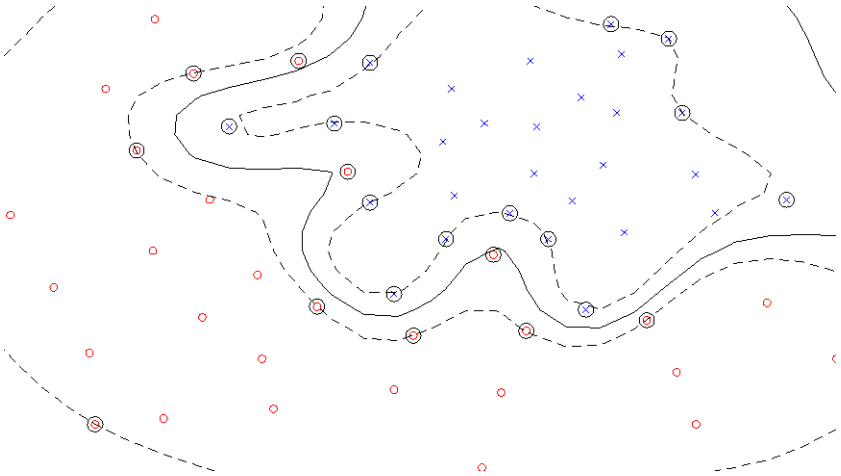
Overfitting

- Both the kernel *k* (type and parameter) and penalty *C* are problem dependent and need to be determined by user.
- Simple model has better generalization to unseen data, complex model can be overfitted to training data.
- However, simple model may exhibit a limited separation (underfitting).

Linear kernel (underfit in this example)

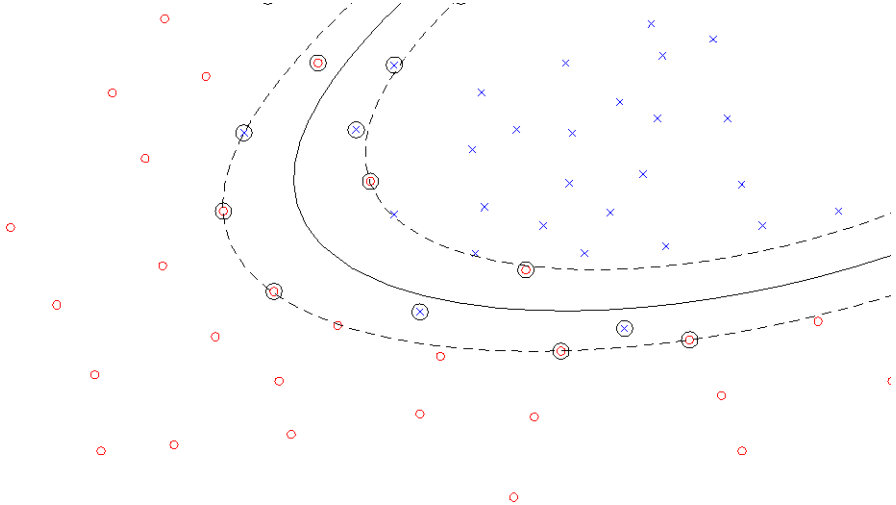


RBF kernel, σ=0.2 (overfit)

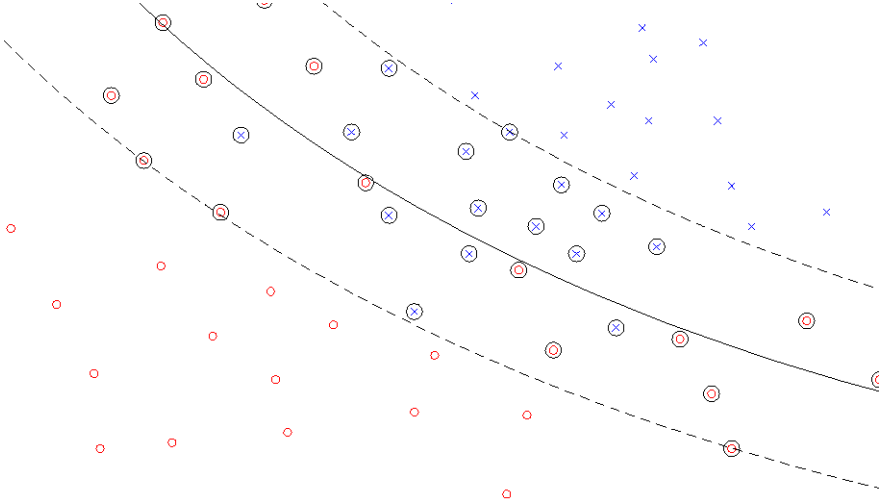


Overfitting

RBF kernel, σ =1.0

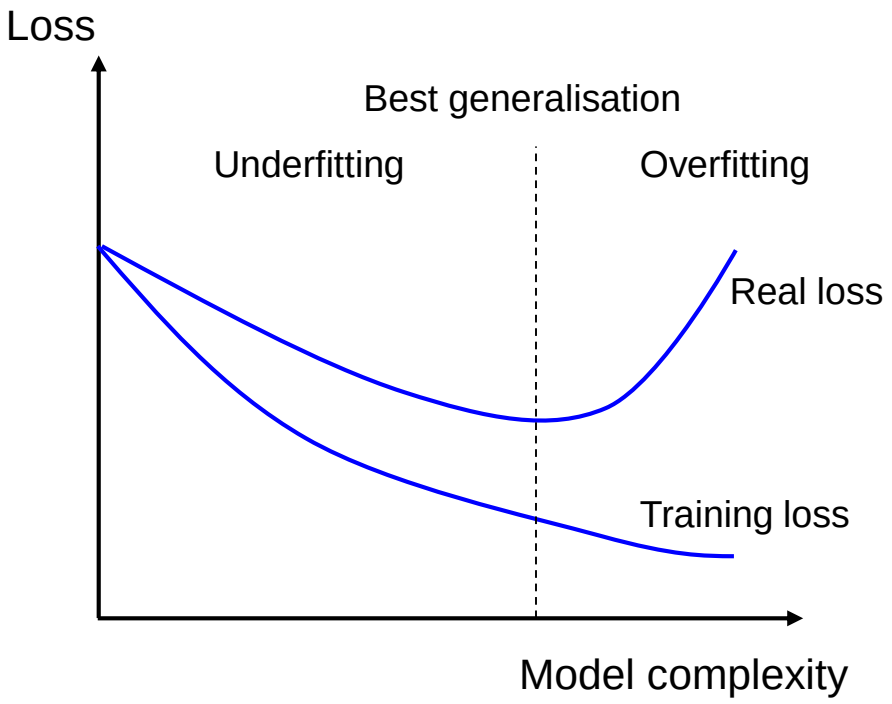


RBF kernel, σ =5.0
(underfit)



Overfitting

- As complexity increases, the model overfits the data:
  - Training loss (classification error of training data) decreases.
  - Real loss (classification error of testing data) increases.
- We need to penalize model complexity = to generalise.



Toolbox for SVM

Statistical Pattern Recognition Toolbox for Matlab

- http://cmp.felk.cvut.cz/cmp/software/stprtool/

LibSVM

- https://www.csie.ntu.edu.tw/~cjlin/libsvm/

Note labels, kernel, other parameters might be defined in slight different ways in different toolboxes.

- E.g. labels = {1 or -1}, or {1 or 0}.
- Epsilon and Tolerance are set close to zero (they indicate how strict the constraints and conditions should be met).
- Among different optimization techniques, we often adopt SMO (sequential minimal optimization) algorithm.

Multi-class SVM

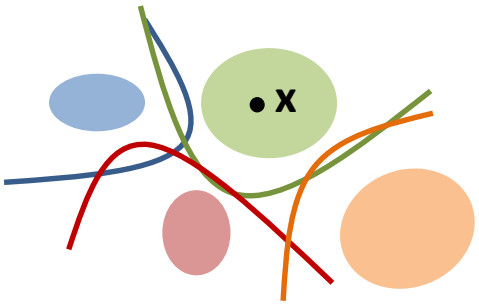
No definitive multi-class SVM formulation.
In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs.

One-versus-the-rest

- Training:
  - Given an *M*-class problem, we train *M* separate SVMs. Each distinguishes images of one category from images of all the other *M-1* categories.
  - The *m*-th SVM $y_m(\mathbf{x})$ is trained by $\boldsymbol{C}_m$ as the positive class and the remaining *M-1* classes as the negative class.
- Testing:
  - Given a query image **x**, we apply each SVM to the query and assign to it the class of the SVM that returns the highest SVM output value

$$y(\mathbf{x}) = \max_m y_m(\mathbf{x})$$

- Issues:
  - The output values $y_m(\mathbf{x})$ for different classifiers have no appropriate scales.
  - The training data sets are imbalanced.



SVM output normalisation

SVM output score $y(\mathbf{x})$ can be normalised to [0,1] either by

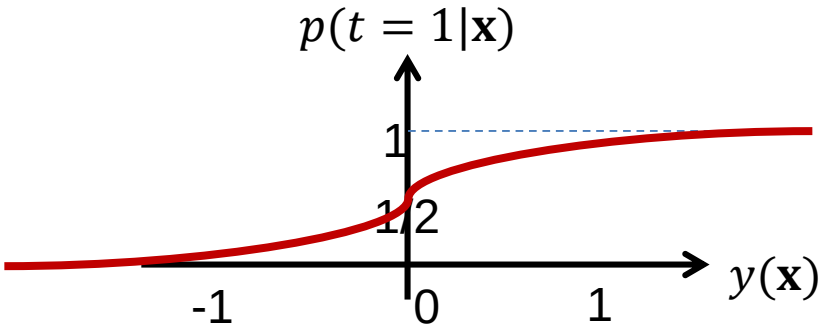
$$(y(\mathbf{x}) - \min(y(\mathbf{x})))/(\max(y(\mathbf{x})) - \min(y(\mathbf{x})))$$

or fitting a logistic sigmoid to the output of SVM:

- The conditional probability takes the form of

$$p(t = 1|\mathbf{x}) = S(Ay(\mathbf{x}) + B)$$

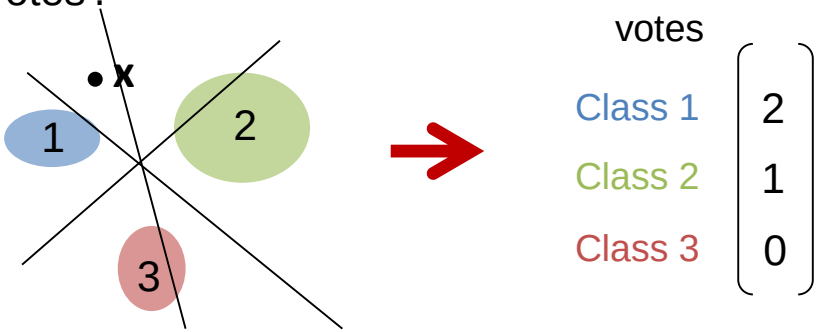
where *A, B* are fixed parameters and *S* is the sigmoid function, $S(z) = \frac{1}{1 + e^{-z}}$



Multi-class SVM

One-versus-one

- Training:
  - Given an *M*-class problem, we train *M(M-1)/2* separate SVMs.
  - We take all possible pairs of classes.
  - An SVM is learnt for each pair of classes, i.e. C_i as the positive class and C_j ,j≠i as the negative class.
- Testing:
  - Each learned SVM votes for a class to assign to a query image.
  - Classification of the query image is by assigning the class has the highest number of 'votes'.



- Issues:
  - It requires a large number of SVMs. Training and testing time depends on the complexity of individual SVMs.
  - A fuzzy case happens if multiple classes receive an equal number of votes.

SVMs pros and cons

Pros

- Many publicly available SVM packages: http://www.kernel-machines.org/software
- Kernel-based framework is very powerful, flexible
- SVMs work very well in practice, even with small training sample sizes

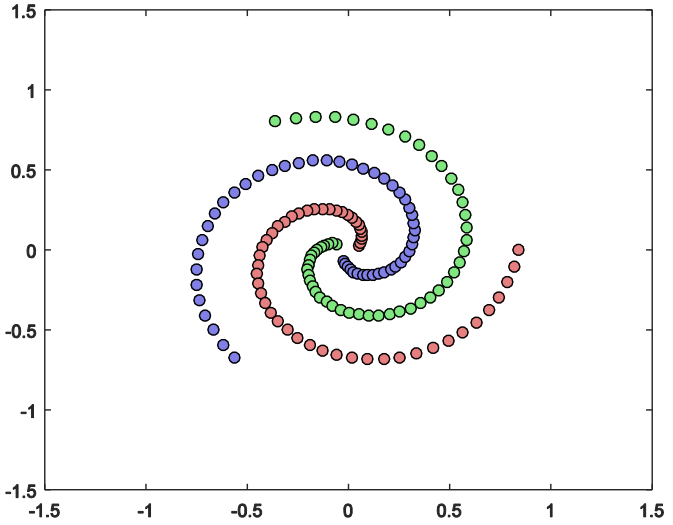
Cons

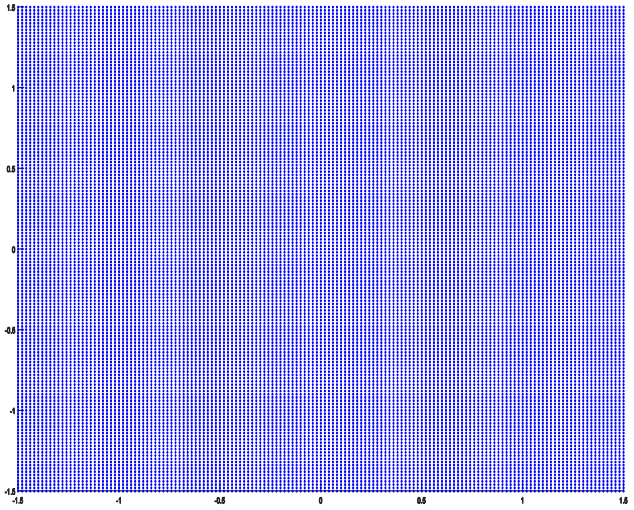
- No direct multi-class SVM, must combine two-class SVMs
- Computation and memory in training
  - During training time, must compute matrix of kernel values for every pair of examples
  - Learning can take a very long time for large-scale problems
- Computational cost of nonlinear SVM classification is very high: $O(KxN_{sv})$

Etc.

Toy Spiral data

- Training
  - data_train(:,1:2) : [num_data x dim] Training 2D vectors
  - data_train(:,3) : [num_data x 1] Label of training data, {1,2,3}
- Testing
  - data_test(:,1:2) : [num_data x dim] Testing 2D vectors, 2D points in the uniform dense grid within the range of [-1.5, 1.5]
  - data_test(:,3) : N/A





Etc.

We can use different performance measures to evaluate our multi-class classifiers:

- overall classification error rate (or the recognition rate)
- confusion matrix
- mean ranks
  - These are the mean position of the correct labels, when labels output by the multi-class classifier are sorted by the classifier score or the similar.
  - E.g. suppose we have the following three sample queries for a system that tries to translate English words to their plurals. In each case, the system makes three guesses, with the first one being the one it thinks is most likely correct:

Query	Results	Correct response	Rank
cat	catten, cati, **cats**	cats	3
torus	torii, **tori**, toruses	tori	2
virus	**viruses**, virii, viri	viruses	1

Confusion matrix