COMP9444: Neural Networks

Support Vector Machines

Optimal Hyperplane

Suppose the training data

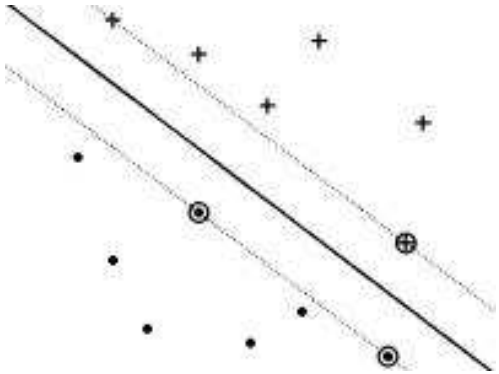
$$(\vec{x}_1, y_1), ..., (\vec{x}_N, y_N),\ \vec{x}_i \in \mathrm{R}^m,\ y_i \in \{-1, +1\}$$

can be separated by a hyperplane

$$\vec{w}^{\mathrm{T}}\vec{x} + b = 0.$$

The hyperplane which separates the training data without error and has maximal distance to the closest training vector is called the Optimal Hyperplane.

Support Vectors



The Optimal separating hyperplane has maximal margin to the training examples. Those lying on the boundary are called the Support Vectors.

Optimal Hyperplane

The aim is to find the hyperplane with the maximum geometric margin, in the input vector space. For any point the functional margin is given by:

$$\hat{\rho}_i = y_i(\vec{w}^T\vec{x}_i + b)$$

The geometric margin can be determined from the functional margin

$$\rho_i = y_i\left(\frac{\vec{w}}{\|\vec{w}\|}^T\vec{x}_i + \frac{b}{\|\vec{w}\|}\right)$$

$$\rho_i = \frac{\hat{\rho}_i}{\|\vec{w}\|}$$

The optimal hyperplane is the minimal margin over all points, $\rho = \min_i \rho_i$.

Optimal Hyperplane

The optimal hyperplane can be found according to:

$$\max_{\hat{\rho},\vec{w},b} \frac{\hat{\rho}}{\|\vec{w}\|}$$

based on the constraint:

$$y_i(\vec{w}^T\vec{x}_i + b) \geq \hat{\rho}$$

Rewriting, and scaling the parameters so that $\hat{\rho} = 1$ gives us:

$$\min_{w,b} \frac{1}{2}\vec{w}^T\vec{w} \text{ , such that: } y_i(\vec{w}^T\vec{x}_i + b) - 1 \geq 0$$

Optimal Hyperplane

The solution can be found using the method of Lagrange multipliers, with the KKT inequality constraint:

$$y_i(\vec{w}^T x_i + b) - 1 \leq 0$$

as follows:

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\vec{w}^T\vec{w} + \sum_i^N \alpha_i(y_i(\vec{w}^T x_i + b) - 1)$$

The solution for $\vec{w}, b$ is found according to:

$$\max_{\vec{\alpha},\vec{\alpha}\geq 0} \min_{\vec{w},b} \mathcal{L}(\vec{w}, b, \vec{\alpha})$$

Lagrangian Formulation

The parameters α_i are known as Lagrange multipliers. The point $(\vec{w}, b, \vec{\alpha})$ is the critical point of the Lagrangian function

$$\begin{aligned} L(\vec{w}, b, \vec{\alpha}) &= \frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} - \sum_{i=1}^N \alpha_i [y_i(\vec{w}^{\mathrm{T}}\vec{x}_i + b) - 1] \\ &= \frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} - \vec{w}^{\mathrm{T}}\Big(\sum_{i=1}^N \alpha_i y_i \vec{x}_i\Big) - b\sum_{i=1}^N \alpha_i y_i + \sum_{i=1}^N \alpha_i \\ \mathcal{L}(\vec{\alpha}) &= \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \vec{x}_i^{\mathrm{T}}\vec{x}_j \end{aligned}$$

There is a duality theorem which allows us to eliminate $\vec{w}$ and b and compute the α's directly from the above equation. Notice that the actual data points $\vec{x}_i$ appear only in the form of dot products $\vec{x}_i^{\mathrm{T}}\vec{x}_j$.

Primal Problem

The solution can be found according to: $\max_{\vec{\alpha}} \mathcal{L}(\vec{\alpha})$

subject to the constraints:

$$\sum_{i=1}^N \alpha_i y_i = 0, \quad (1)$$

$$\alpha_i \geq 0 \quad \text{for} \quad i = 1, \ldots N \quad (2)$$

Note: in the more general case where the points are not linearly separable, but we want to find the best fit we can, condition (2) will be replaced by

$$0 \leq \alpha_i \leq C,$$

where C is a regularization parameter which can be determined experimentally or analytically, using estimate of VC-dim.

Geometrical Considerations

The support vectors are the ones for which the constraint is "tight", i.e.

$$y_i(\vec{w}^{\mathrm{T}}\vec{x}_i + b) - 1 = 0$$

The other data points, i.e. those for which the point $(\vec{w}, b)$ does not lie on the constraint boundary, must have $\alpha_i = 0$. Therefore, the following product must be zero for all i

$$\alpha_i\,[y_i(\vec{w}^{\mathrm{T}}\vec{x}_i + b) - 1] = 0$$

Note also that

$$\vec{w}^{\mathrm{T}}\vec{w} = \vec{w}^{\mathrm{T}}\Big(\sum_{i=1}^{N}\alpha_i y_i \vec{x}_i\Big) = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i^{\mathrm{T}}\vec{x}_j$$

Finding the Optimal Hyperplane

The optimal values for $\vec{\alpha}$ can be found by solving the Primal Problem with quadratic programming techniques.

$\vec{w}$ can then be found from

$$\vec{w} = \sum_{i=1}^{N}\alpha_i y_i \vec{x}_i$$

b can be found from any of the support vectors (those for which $\alpha_i \neq 0$) by

$$b = y_i - \vec{w}^{\mathrm{T}}\vec{x}_i$$

VC-dim of Optimal Hyperplanes

Consider the following set of training vectors $X^* = \{\vec{x}_1, ..., \vec{x}_N\}$, bounded by a sphere of radius D, i.e.

$$|\vec{x}_i - \vec{a}| \leq D,\ x_i \in X^*,$$

where $\vec{a}$ is the centre of the sphere.

Theorem (Vapnik, 1995) A subset of canonical hyperplanes

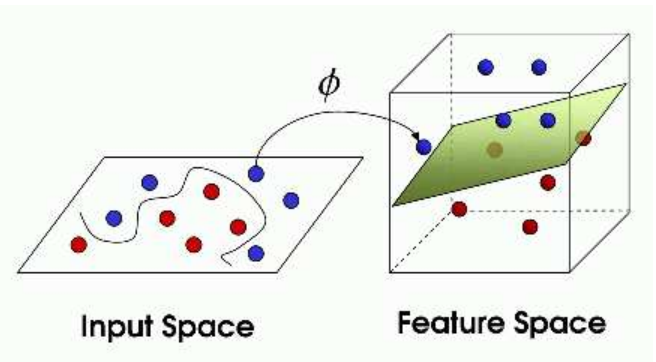
$$f(\vec{x}, \vec{w}, b) = \mathrm{sign}((\vec{w}^{\mathrm{T}}\vec{x}) + b),$$

defined on X^* and satisfying the constraint $||\vec{w}|| \leq A$ has VC-dimension h bounded by the inequality

$$h \leq \min([D^2A^2], m) + 1,$$

m being the number of dimensions.

Kernel Methods



To address non-linear relationships, a Support Vector Machine maps the input space (nonlinearly) into a high-dimensional feature space, and then constructs an Optimal Hyperplane in the feature space.

Support Vector Machines

Two problems:

- How to find a separating hyperplane that will generalise well?

  The dimensionality of the feature space will be very large. As a consequence, not all separating hyperplanes will generalise well.

- How to treat computationally such high-dimensional spaces?

  A very high-dimensional feature space cannot be explicitly computed. e.g. considering polynomials of degree 4 or 5 in a 200 dimensional input space results in a billion dimensional feature space. Obviously, "special" treatment of such spaces is required.

The Kernel Trick

The features $(\phi_1(\vec{x}), \ldots, \phi_m(\vec{x}))$ will play the same role that was previously played by the co-ordinates of $\vec{x}$. By convention, we add an additional "bias" feature $\phi_0()$ with $\phi_0(\vec{x}) = 1$ for all $\vec{x}$. Recall that we only need to be able to compute the dot products

$$K(\vec{x}_i, \vec{x}_j) = \vec{\phi}^{\,T}(\vec{x}_i)\,\vec{\phi}\,(\vec{x}_j) = \sum_{k=0}^{m} \phi_k(\vec{x}_i)\phi_k(\vec{x}_j).$$

Fortunately $K(\vec{x}_i, \vec{x}_j)$, which is known as a Kernel function, can often be computed directly without having to compute the individual ϕ_k's. This is the key idea which makes Support Vector Machines computable.

Polynomial Features

For example, if $\vec{\phi}()$ is the set of polynomials in $x_1, \ldots, x_l$ of degree up to p, then

$$K(\vec{x}, \vec{y}) = (1 + \vec{x}^T \vec{y})^p$$

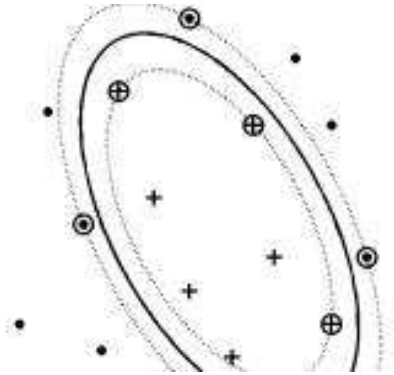
e.g. if $l = 2$ and $p = 2$,

$$\begin{aligned} K(\vec{x}, \vec{y}) &= (1 + x_1 y_1 + x_2 y_2)^2 \\ &= 1 + 2x_1 y_1 + 2x_2 y_2 + x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2 \\ &= \sum_{k=0}^{m} \phi_k(\vec{x})\phi_k(\vec{y}) = \vec{\phi}^{\,T}(\vec{x})\,\vec{\phi}\,(\vec{y}) \end{aligned}$$

where

$$\vec{\phi}(\vec{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)$$

Example



Example of an SVM using a quadratic feature space.

Other Popular Kernel Functions

Radial Basis Function machine:

$$K(\vec{x},\vec{y}) = \exp(-\frac{||\vec{x}-\vec{y}||^2}{2\sigma^2})$$

Two-layer perceptron machine:

$$K(\vec{x},\vec{y}) = \tanh(\,c(\vec{x}^{\mathrm{T}}\vec{y}) - b\,)$$

These kernels are not quite well behaved from the point-of-view of theoretical mathematics (in fact, the RBF kernel corresponds to an infinite-dimensional feature space). Despite this, they tend to give good performance in practice.

The tanh kernel can be seen as an alternative to the backpropagation algorithm, which additionally tells you the number of hidden nodes (equal to the number of support vectors).

Support Vector Machines

Theorem (Vapnik, 1995)

If the training vectors are separated by the Optimal hyperplane, then the expectation value of the probability of committing an error on a test example is bounded by the ratio of the expectation of the number of support vectors to the number of examples in the training set:

$$E[P(error)] \leq \frac{E[\text{number of support vectors}]}{(\text{number of training vectors}) - 1}$$

It is interesting to note that this bound neither depends on the dimensionality of the feature space (nor on the norm of the vector of coefficients, nor on the bound of the norm of the input vectors).

Support Vector Machines

Handwritten digit recognition with benchmark training and test data from the US Postal Service.

16×16 input values.

For each of the 10 classes an individual classifier was learned. There were 7,300 training patterns and 2,000 test patterns.

Classifier	Raw error %
Human performance	2.5
Decision tree learner; C4.5	16.2
Best two-layer neural network	5.9
Five-layer network (LeNet 1)	5.1

Support Vector Machines

Polynomials up to degree 7 were used for Support Vector Machines.

degree of polynomial	dimensionality of feature space	support vectors	raw error
1	256	282	8.9
2	$\approx$ 33,000	227	4.7
3	$\approx 10^6$	274	4.0
4	$\approx 10^9$	321	4.2
5	$\approx 10^{12}$	374	4.3
6	$\approx 10^{14}$	377	4.5
7	$\approx 10^{16}$	422	4.5

Support Vector Machines

	Chosen Classifier			Number of test errors						
Digit	degree	dimensions	h_{est}	1	2	3	4	5	6	7
0	3	$\approx 10^6$	530	36	14	11	11	11	12	17
1	7	$\approx 10^{16}$	101	17	15	14	11	10	10	10
2	3	$\approx 10^6$	842	53	32	28	26	28	27	32
3	3	$\approx 10^6$	1157	57	25	22	22	22	22	23
4	4	$\approx 10^9$	962	50	32	32	30	30	29	33
5	3	$\approx 10^6$	1090	37	20	22	24	24	26	28
6	4	$\approx 10^9$	626	23	12	12	15	17	17	19
7	5	$\approx 10^{12}$	530	25	15	12	10	11	13	14
8	4	$\approx 10^9$	1445	71	33	28	24	28	32	44
9	5	$\approx 10^{12}$	1226	51	18	15	11	11	12	15

Summary

- The Support Vector Machine (SVM) is an elegant and highly principled learning method which selects relevant features from a very high dimensional feature space.
- SVM's use quadratic programming techniques to solve a constrained optimization problem, thus avoiding the need to deal directly with the high dimensional feature space.
- SVM's run only in batch mode, can't be trained incrementally.
- Training times for SVM's are slower than backpropagation, because there is no control over the number of potential support vectors, and there is no natural way to incorporate prior knowledge about the task.