COMP90049 Knowledge Technologies

KNN and Support Vector Machines (LectureSet7) 2017

Rao Kotagiri
Department of Computing and Information Systems
The Melbourne School of Engineering

Some of slides are derived from Prof Vipin Kumar and modified, http://www-users.cs.umn.edu/~kumar/

K- Nearest Neighbour (KNN)

•It is a geometric based method.

•It is a simple technique and works well in many situations.

•There is no learning involved.

•Given a training data set with labelled information and test case we find K training data points (records) nearest to the test case.

•The test case is labelled with the label of the most frequent label of the K nearest points to the test case.

K- Nearest Neighbour (KNN) method

For K =3, the example shows how KNN labels an unknown test case △.

Some difficulties with KNN:

• What should be the distance function?

• How do we efficiently find the K-nearest neighbouring points to the test case?

• As the number of dimensions increases all the points seem to be more or less at an equal distance (curse of dimensionality). In this case KNN performs poorly and in fact most machine learning methods can fail!

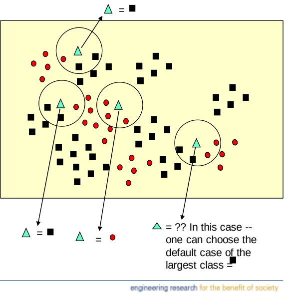
Advantages:

• Very simple scheme with no learning phase.

• Works well for small dimensional data.

• New data can be easily added and hence it is a Lazy Method.

• No need to estimate any probabilities but the notion of K nearest implicitly models the distribution when determining the K-nearest points



K- Nearest Neighbour (KNN) method

Some difficulties with KNN:

• What should be the distance function?

Let p = <p1,p2,…,pn> and q = <q1,q2,…,qn> two points in an n-dimensional space.

$$\text{Euclidian distance }(p,q) = \|p-q\|^2 = \sqrt{\sum_{i=1}^{n}(pi-qi)^2}$$

$$\text{Minkowski distance}(p,q,k) = \sqrt[k]{\sum_{i=1}^{n}|pi-qi|^k}$$

$$Radial\_Basis\_dist(\mathbf{p},\mathbf{q},\sigma) = \exp\left(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2\sigma^2}\right)$$

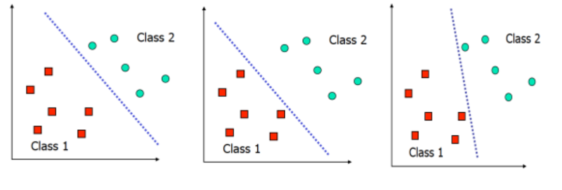
When
$k = 1$ *is called* $L1$ *Norm*
$k = 2$ *is called* $L2$ *Norm*

$$Cos\_dist(p,q) = \frac{p.q}{\|p\| * \|q\|}$$

Support Vector Machines (SVM)

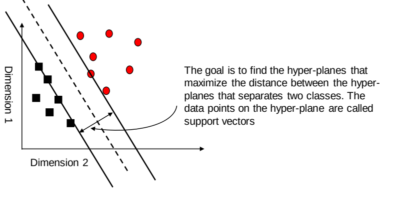
We can separate a linearly separable two-class using many decision boundaries. However, some decision boundaries are far better than the others. The goal of SVM is to find the optimal decision boundary.

Examples of a good Decision Boundary — Examples of Bad Decision Boundaries



Support Vector Machines

Support vector machines is like Naïve Bayesian Classifier is a statistical machine learning Technique and it is also a geometric based method like KNN (K-nearest neighbour)



COMP90049 Knowledge Technologies

Support Vector Machines

Training Data					
i	X				Y
1	X_{11}	X_{12}	…	X_{1m}	+1
2	X_{21}	X_{22}	…	X_{2m}	- 1
…					
n	X_{n1}	X_{n2}		X_{nm}	-1

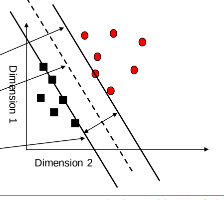
$X_i = \langle X_{i1}, X_{i2}, \ldots, X_{im} \rangle$ these are the feature values of data instance i
Y_i = label of instance i. X_{ik} is real number.

The training set can be represented as

$$\{(\mathrm{x}_i, y_i) : i = 1, \ldots, n\} \subset \Re^n \times \{1, -1\}$$

The hyper-planes can be represented as

$$\mathbf{w}^T \mathrm{x}_i + b = +1$$
$$\mathbf{w}^T \mathrm{x}_i + b = 0$$
$$\mathbf{w}^T \mathrm{x}_i + b = -1$$



COMP90049 Knowledge Technologies

Support Vector Machines

Training Data					
i	X				Y
1	X_{11}	X_{12}	…	X_{1m}	+1
2	X_{21}	X_{22}	…	X_{2m}	- 1
…					
n	X_{n1}	X_{n2}		X_{nm}	-1

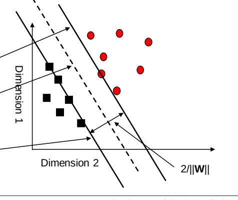
$X_i = \langle X_{i1}, X_{i2}, \ldots, X_{im} \rangle$ these are the feature values of data instance i
Y_i = label of instance i. X_{ik} is real number.

The training set can be represented as

$$\{(\mathrm{x}_i, y_i) : i = 1, \ldots, n\} \subset \Re^n \times \{1, -1\}$$

The hyper-plane can be represented as

$$\mathbf{w}^T \mathrm{x}_i + b = +1$$
$$\mathbf{w}^T \mathrm{x}_i + b = 0$$
$$\mathbf{w}^T \mathrm{x}_i + b = -1$$



COMP90049 Knowledge Technologies

Separating Hyper-plane

For dataset $\{x_i\}$:

$$f(\mathrm{x}_i) = \mathbf{w}^T \mathrm{x}_i + b = \begin{cases} >= 1, & if\ y_i = 1; \\ <= -1, & if\ y_i = -1. \end{cases}$$

COMP90049 Knowledge Technologies

Separating Hyper-plane

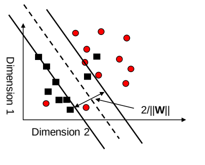
For dataset $\{x_i\}$:

$$f(\mathrm{x}_i) = \mathbf{w}^T \mathrm{x}_i + b = \begin{cases} \geq 1, & if\ y_i = 1; \\ \leq -1, & if\ y_i = -1. \end{cases}$$

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

Unfortunately we may not find hyper-panes that can separate given set of training instances. Therefore we allow some margin of errors

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$



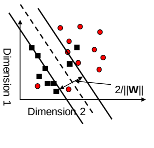
COMP90049 Knowledge Technologies

Lagrange function

The optimal hyper-plane is found as the solution to the optimization problem:

minimize $\tau(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$

subject to $y_i(\mathbf{w}^T \mathrm{x}_i + b) \geq 1 - \xi_i \quad i = 1..n$



Lagrangian function:

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}^T \mathrm{x}_i + b) - 1 + \xi_i)$$

COMP90049 Knowledge Technologies

Lagrangian function

For data in the input space:

maximize $Q(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$

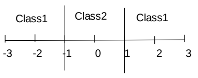
subject to $\sum_{i=1}^{n}\alpha_i y_i = 0 \qquad 0 \leq \alpha_i \leq C$

Once optimal $a_1, a_2, \ldots, a_n$ are determined, w can be computed as

$$W = \sum_{i=1}^{n}\alpha_i y_i x_i$$

Kernel mapping

x	y
-3	c1
-2	c1
-1	c2
0	c2
+1	c2
+2	c1
+3	c1



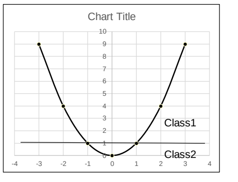
We cannot separate the points by drawing a boundary.

Kernel mapping

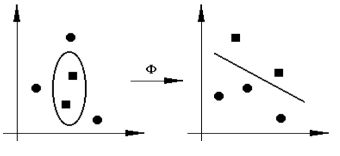
x	y
-3	c1
-2	c1
-1	c2
0	c2
+1	c2
+2	c1
+3	c1

$\Rightarrow$

x	x^2	y
-3	9	c1
-2	4	c1
-1	1	c2
0	0	c2
+1	1	c2
+2	4	c1
+3	9	c1

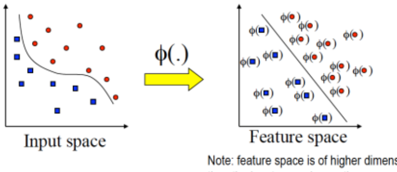


Kernel mapping



$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\Phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\Phi}(\boldsymbol{x}_j)$$

Kernel mapping



Note: feature space is of higher dimension than the input space in practice

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\Phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\Phi}(\boldsymbol{x}_j)$$

Non-linear case

For data in the input space:

minimize $\quad Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathrm{x}_i, \mathrm{x}_j)$

subject to $\quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad 0 \le \alpha_i \le C$

where K is a kernel function, by which SVMs may construct a better optimal separating hyper-plane into a higher dimensional feature space.

Kernel functions

Linear kernel: $\quad K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \cdot \boldsymbol{x}_j$

Polynomial kernel: $\quad K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \cdot \boldsymbol{x}_j + \theta)^d$

Radial-basis function (RBF) kernel:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$$

Weaknesses of SVMs

- Best performance depends on the choice of the kernel and its parameters.
- Learning is very expensive.
- Generalization to multi-class memberships needs several SVMs

Some solutions

Build one versus the rest:

- E.g. 3 class problem c1, c2 and c3
- C1 vs C2 or C3 - SVM1
- C2 vs C1or C3 - SVM2
- C3 vs C1 or C2 - SVM3

Requires n classifiers

SVM1 C1 vs C2C3	SVM2 C2 vs C1C3	SVM3 C3 vs C1C2	Final Class
C1	C2	c3	Largest-class-of (C1,C2,C3)
C1	C2	C1 or C2	Largest-class-of (C1,C2)
C1	C1 or C3	c3	Largest-class-of (C1,C3)
C1	C1 or C3	C1 or C2	C1
C2 or C3	C2	C3	Largest-class-of (C2,C3)
C2 or C3	C2	C1 or C2	C2
C2 or C3	C1 or C3	C3	C3
C2 or C3	C1 or C3	C1 or C3	Largest-class-of (C1,C2,C3)

Weaknesses of SVMs

•Generalization to multi-class memberships

Some solutions

- Build one versus the rest:
- One Versus one

E.g. 3 class problem C1,C2 and C3

- C1 vs C2 -SVM1
- C1 vs C3 -SVM2
- C2 vs C3 -SVM3

SVM1 C1 vs C2	SVM2 C1 vs C3	SVM3 C2 vs C3	Final Class
C1	C1	C2	C1
C1	C1	C3	C1
C1	C3	C2	Largest-Class-of (C1,C2,C3)
C1	C3	C3	C3
C2	C1	C2	C2
C2	C1	C3	Largest-Class-of C1,C2,C3)
C2	C3	C2	C2
C2	C3	C3	C3

- Requires building $\binom{n}{2}$ classifiers!

Weaknesses of SVMs

•Generalization to multi-class memberships

Some solutions

- Build one versus the rest:
- One Versus one
- Build log(n) binary classifiers

Example assume we have 4 classes

- We build log(4) = 2 classifies

Training Data					
i	X				Class
1	X11	X12	…	X1m	C3
2	X21	X22	…	X2m	C2
3	X31	X32	…	X3m	C1
…					
n	Xn1	Xn2		Xnm	C4

Y1	Y2	Class
-1	-1	C1
-1	+1	C2
+1	-1	C3
+1	+1	C4

Weaknesses of SVMs

Y1	Y2	Class
-1	-1	C1
-1	+1	C2
+1	-1	C3
+1	+1	C4

Training Data					
i	X				Class
1	X11	X12	…	X1m	C3
2	X21	X22	…	X2m	C2
3	X31	X32	…	X3m	C1
…					
n	Xn1	Xn2		Xnm	C4

Training Data						
	X				Y1	Y2
1	X11	X12	…	X1m	+1	-1
2	X21	X22	…	X2m	- 1	+1
3	X31	X32	…	X3m	- 1	-1
…						
n	Xn1	Xn2		Xnm	+1	+1

Weaknesses of SVMs

Training Data					
i	X				Class
1	X11	X12	…	X1m	C3
2	X21	X22	…	X2m	C2
3	X31	X32	…	X3m	C1
…					
n	Xn1	Xn2		Xnm	C4

Training Data						
	X				Y1	Y2
1	X11	X12	…	X1m	+1	-1
2	X21	X22	…	X2m	- 1	+1
3	X31	X32	…	X3m	- 1	-1
…						
n	Xn1	Xn2		Xnm	+1	+1

Training Data-1					
i	X				Y1
1	X11	X12	…	X1m	+1
2	X21	X22	…	X2m	- 1
3	X31	X32	…	X3m	- 1
…					
n	Xn1	Xn2		Xnm	+1

Training Data-2					
i	X				Y2
1	X11	X12	…	X1m	-1
2	X21	X22	…	X2m	+1
3	X31	X32	…	X3m	-1
…					
n	Xn1	Xn2		Xnm	+1