CS685: Data Mining
Classification

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~cs685/

1^{st} semester, 2012-13

Tue, Wed, Fri 0900-1000 at CS101

Outline

1. Naïve Bayes classifiers
2. Bayesian networks
3. Support vector machines (SVM)

Outline

1. Naïve Bayes classifiers
2. Bayesian networks
3. Support vector machines (SVM)

Bayes' theorem

$$P(C|O) = \frac{P(O|C)P(C)}{P(O)}$$

- $P(C|O)$ is the probability of class C given object O – posterior probability
- $P(O|C)$ is the probability that O is from class C – likelihood probability
- $P(C)$ is the probability of class C – prior probability
- $P(O)$ is the probability of object O – evidence probability

$$posterior = \frac{likelihood \times prior}{evidence}$$

Naïve Bayes classifier

- Naïve Bayes classifier or simple Bayes classifier
- To classify a new object O_q, compute posterior probabilities $P(C_i|O_q)$ for all classes $C_i, i = 1, \ldots, k$

$$P(C_i|O_q) = \frac{P(O_q|C_i)P(C_i)}{P(O_q)}$$

- Bayes decision rule: The class with the *highest* posterior probability is chosen
- $P(O_q)$ is constant for all classes and, therefore, can be removed

Naïve Bayes classifier

- Naïve Bayes classifier or simple Bayes classifier
- To classify a new object O_q, compute posterior probabilities $P(C_i|O_q)$ for all classes $C_i, i = 1, \dots, k$

$$P(C_i|O_q) = \frac{P(O_q|C_i)P(C_i)}{P(O_q)}$$

- Bayes decision rule: The class with the *highest* posterior probability is chosen
- $P(O_q)$ is constant for all classes and, therefore, can be removed
- Since it maximizes posterior probability, it is called maximum a posteriori (MAP) method

Naïve Bayes classifier

- Naïve Bayes classifier or simple Bayes classifier
- To classify a new object O_q, compute posterior probabilities $P(C_i|O_q)$ for all classes $C_i, i = 1, \dots, k$

$$P(C_i|O_q) = \frac{P(O_q|C_i)P(C_i)}{P(O_q)}$$

- Bayes decision rule: The class with the *highest* posterior probability is chosen
- $P(O_q)$ is constant for all classes and, therefore, can be removed
- Since it maximizes posterior probability, it is called maximum a posteriori (MAP) method
- If priors are unknown or same, this essentially maximizes the likelihood $P(O_q|C_i)$
- This is called maximum likelihood (ML) method

Computing likelihood

- In general, O_q has m features $O_{q_1}, \ldots, O_{q_m}$

$$
\begin{aligned}
P(O_q|C_i) &= P(O_{q_1}, O_{q_2}, \ldots, O_{q_m}|C_i) \\
&= P(O_{q_1}|C_i) \times P(O_{q_2}, \ldots, O_{q_m}|O_{q_1}, C_i) \\
&= P(O_{q_1}|C_i) \times P(O_{q_2}|O_{q_1}, C_i) \times P(O_{q_3}, \ldots, O_{q_m}|O_{q_1}, O_{q_2}, C_i)
\end{aligned}
$$

Computing likelihood

- In general, O_q has m features $O_{q_1}, \dots, O_{q_m}$

$$
\begin{aligned}
P(O_q|C_i) &= P(O_{q_1}, O_{q_2}, \dots, O_{q_m}|C_i) \\
&= P(O_{q_1}|C_i) \times P(O_{q_2}, \dots, O_{q_m}|O_{q_1}, C_i) \\
&= P(O_{q_1}|C_i) \times P(O_{q_2}|O_{q_1}, C_i) \times P(O_{q_3}, \dots, O_{q_m}|O_{q_1}, O_{q_2}, C_i)
\end{aligned}
$$

- *Simple* or *naïve* assumption is now applied: All class conditional probabilities are independent

Computing likelihood

- In general, O_q has m features $O_{q_1}, \ldots, O_{q_m}$

$$
\begin{aligned}
P(O_q|C_i) &= P(O_{q_1}, O_{q_2}, \ldots, O_{q_m}|C_i) \\
&= P(O_{q_1}|C_i) \times P(O_{q_2}, \ldots, O_{q_m}|O_{q_1}, C_i) \\
&= P(O_{q_1}|C_i) \times P(O_{q_2}|O_{q_1}, C_i) \times P(O_{q_3}, \ldots, O_{q_m}|O_{q_1}, O_{q_2}, C_i)
\end{aligned}
$$

- *Simple* or *naïve* assumption is now applied: All class conditional probabilities are independent

$$
\begin{aligned}
P(O_{q_j}, O_{q_k}|C_i) &= P(O_{q_j}|C_i) \times P(O_{q_k}|O_{q_j}, C_i) \\
&= P(O_{q_j}|C_i) \times P(O_{q_k}|C_i) \quad [\because O_{q_j}, O_{q_k} \text{ are independent}]
\end{aligned}
$$

Computing likelihood

- In general, O_q has m features $O_{q_1}, \dots, O_{q_m}$

$$\begin{aligned}
P(O_q|C_i) &= P(O_{q_1}, O_{q_2}, \dots, O_{q_m}|C_i) \\
&= P(O_{q_1}|C_i) \times P(O_{q_2}, \dots, O_{q_m}|O_{q_1}, C_i) \\
&= P(O_{q_1}|C_i) \times P(O_{q_2}|O_{q_1}, C_i) \times P(O_{q_3}, \dots, O_{q_m}|O_{q_1}, O_{q_2}, C_i)
\end{aligned}$$

- *Simple* or *naïve* assumption is now applied: All class conditional probabilities are independent

$$\begin{aligned}
P(O_{q_j}, O_{q_k}|C_i) &= P(O_{q_j}|C_i) \times P(O_{q_k}|O_{q_j}, C_i) \\
&= P(O_{q_j}|C_i) \times P(O_{q_k}|C_i) \quad [\because O_{q_j}, O_{q_k} \text{ are independent}]
\end{aligned}$$

$$\begin{aligned}
\therefore P(O_q|C_i) &= P(O_{q_1}, O_{q_2}, \dots, O_{q_m}|C_i) \\
&= \prod_{j=1}^{m} P(O_{q_j}|C_i)
\end{aligned}$$

Training

- How to estimate $P(O_{q_j}|C_i)$

Training

- How to estimate $P(O_{q_j}|C_i)$
- Examine all training objects pertaining to class C_i

Training

- How to estimate $P(O_{q_j}|C_i)$
- Examine all training objects pertaining to class C_i
- If O_{q_j} is categorical, then empirical probabilities are the estimates

$$P(O_{q_j} = v|C_i) = \frac{|\{O_k \in C_i : O_{k_j} = v\}|}{|\{O_k \in C_i\}|}$$

- A particular discrete distribution can also be assumed

Training

- How to estimate $P(O_{q_j}|C_i)$
- Examine all training objects pertaining to class C_i
- If O_{q_j} is categorical, then empirical probabilities are the estimates

$$P(O_{q_j} = v|C_i) = \frac{|\{O_k \in C_i : O_{k_j} = v\}|}{|\{O_k \in C_i\}|}$$

- A particular discrete distribution can also be assumed
- If O_{q_j} is numerical, then a certain continuous distribution is assumed
- Generally, Gaussian or normal distribution $N(\mu, \sigma)$
- μ and σ are estimated from training objects in C_i

$$P(O_{q_j} = v|C_i) = N(v; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(v-\mu)^2}{2\sigma^2}}$$

Training

- How to estimate $P(O_{q_j}|C_i)$
- Examine all training objects pertaining to class C_i
- If O_{q_j} is categorical, then empirical probabilities are the estimates

$$P(O_{q_j} = v|C_i) = \frac{|\{O_k \in C_i : O_{k_j} = v\}|}{|\{O_k \in C_i\}|}$$

- A particular discrete distribution can also be assumed
- If O_{q_j} is numerical, then a certain continuous distribution is assumed
- Generally, Gaussian or normal distribution $N(\mu, \sigma)$
- μ and σ are estimated from training objects in C_i

$$P(O_{q_j} = v|C_i) = N(v; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(v-\mu)^2}{2\sigma^2}}$$

- $P(C_i)$ is just the empirical estimate $|C_i|/|D|$

Example: training

Class	Rank	Motivated	Exam marks
Successful (S)	2	Y	78.3
	99	Y	70.3
	5	N	88.5
	87	Y	75.1
Unsuccessful (U)	1	N	76.3
	90	N	66.2
	9	Y	68.1
	62	N	75.4

Example: training

Class	Rank	Motivated	Exam marks
Successful (S)	2	Y	78.3
	99	Y	70.3
	5	N	88.5
	87	Y	75.1
Unsuccessful (U)	1	N	76.3
	90	N	66.2
	9	Y	68.1
	62	N	75.4

Likelihoods

Class	Rank	Motivated	Exam marks
S	$\mu = 48.25$ $\sigma = 51.92$	$P(Y) = 0.75$ $P(N) = 0.25$	$\mu = 78.05$ $\sigma = 7.70$
U	$\mu = 40.50$ $\sigma = 42.68$	$P(Y) = 0.25$ $P(N) = 0.75$	$\mu = 71.50$ $\sigma = 5.10$

Example: testing

- $O_q = (70, Y, 67.3)$

Example: testing

- $O_q = (70, Y, 67.3)$

$$
\begin{aligned}
P(O_q|S) &= P(70|S) \times P(Y|S) \times P(67.3|S) \times P(S) \\
&= N(70; 48.25, 51.92) \times 0.75 \times N(67.3; 78.05, 7.70) \times 0.5 \\
&= 0.00704 \times 0.75 \times 0.0195 \times 0.5 \\
&= 5.16 \times 10^{-5} \\
P(O_q|U) &= P(70|U) \times P(Y|U) \times P(67.3|U) \times P(U) \\
&= N(70; 40.50, 42.68) \times 0.25 \times N(67.3; 71.50, 5.10) \times 0.5 \\
&= 0.00736 \times 0.25 \times 0.0597 \times 0.5 \\
&= 5.49 \times 10^{-5}
\end{aligned}
$$

- Therefore, O_q is from class U

Discussion

- If an estimated probability $P(O_{q_j}|C_i)$ becomes zero, the whole likelihood becomes zero
- Laplacian correction or Laplacian estimation: Add a small ϵ

Discussion

- If an estimated probability $P(O_{q_j}|C_i)$ becomes zero, the whole likelihood becomes zero
- Laplacian correction or Laplacian estimation: Add a small ϵ
- Advantages

Discussion

- If an estimated probability $P(O_{q_j}|C_i)$ becomes zero, the whole likelihood becomes zero
- Laplacian correction or Laplacian estimation: Add a small ϵ
- Advantages
  - Incremental

Discussion

- If an estimated probability $P(O_{q_j}|C_i)$ becomes zero, the whole likelihood becomes zero
- Laplacian correction or Laplacian estimation: Add a small ϵ
- Advantages
  - Incremental
  - Robust to noise as probability of noise is low

Discussion

- If an estimated probability $P(O_{q_j}|C_i)$ becomes zero, the whole likelihood becomes zero
- Laplacian correction or Laplacian estimation: Add a small ϵ
- Advantages
  - Incremental
  - Robust to noise as probability of noise is low
  - Robust to irrelevant attributes as their probability tends to be uniform across classes
- Disadvantages

Discussion

- If an estimated probability $P(O_{q_j}|C_i)$ becomes zero, the whole likelihood becomes zero
- Laplacian correction or Laplacian estimation: Add a small ϵ
- Advantages
  - Incremental
  - Robust to noise as probability of noise is low
  - Robust to irrelevant attributes as their probability tends to be uniform across classes
- Disadvantages
  - Treats attributes as independent and ignores any correlation information

Outline

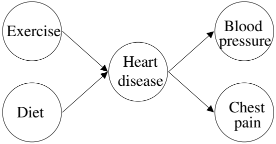
Bayesian networks

- Bayesian networks or Bayesian belief networks or Bayes nets or belief nets
- Takes into account the correlations of attributes by modeling them as conditional probabilities
- Forms a directed acyclic graph (DAG)
- Edges model the *dependencies*
- Parent is the *cause* and children are the *effects*

Bayesian networks

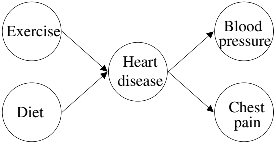
- Bayesian networks or Bayesian belief networks or Bayes nets or belief nets
- Takes into account the correlations of attributes by modeling them as conditional probabilities
- Forms a directed acyclic graph (DAG)
- Edges model the *dependencies*
- Parent is the *cause* and children are the *effects*
- A node is conditionally independent of all its non-descendants given its parents
- For every node, there is a conditional probability table (CPT) that describes its values given its parents' values
- CPT for node X is of the form $P(X|parents(X))$

Example



- CPTs: rows are values; columns are parents (i.e., conditionals)
- Last rows can be inferred, and therefore, omitted

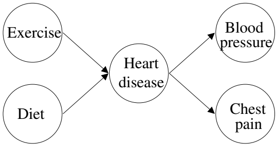
Example



- CPTs: rows are values; columns are parents (i.e., conditionals)
- Last rows can be inferred, and therefore, omitted

Exercise (E)	Φ
regular (r)	0.70
irregular (i)	0.30

Example

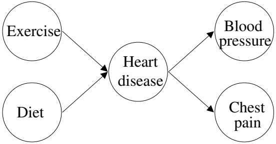


- CPTs: rows are values; columns are parents (i.e., conditionals)
- Last rows can be inferred, and therefore, omitted

Exercise (E)	Φ
regular (r)	0.70
irregular (i)	0.30

Diet (D)	Φ
healthy (h)	0.25
unhealthy (u)	0.75

Example



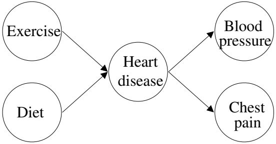
- CPTs: rows are values; columns are parents (i.e., conditionals)
- Last rows can be inferred, and therefore, omitted

Exercise (E)	Φ
regular (r)	0.70
irregular (i)	0.30

Diet (D)	Φ
healthy (h)	0.25
unhealthy (u)	0.75

Heart disease (H)	E=r, D=h	E=r, D=u	E=i, D=h	E=i, D=u
yes (y)	0.25	0.40	0.55	0.80
no (n)	0.75	0.60	0.45	0.20

Example



- CPTs: rows are values; columns are parents (i.e., conditionals)
- Last rows can be inferred, and therefore, omitted

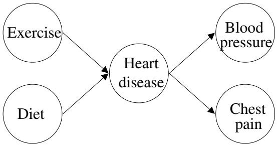
Exercise (E)	Φ
regular (r)	0.70
irregular (i)	0.30

Diet (D)	Φ
healthy (h)	0.25
unhealthy (u)	0.75

Heart disease (H)	E=r, D=h	E=r, D=u	E=i, D=h	E=i, D=u
yes (y)	0.25	0.40	0.55	0.80
no (n)	0.75	0.60	0.45	0.20

Blood pressure (B)	H=y	H=n
normal (l)	0.15	0.80
high (g)	0.85	0.20

Example



- CPTs: rows are values; columns are parents (i.e., conditionals)
- Last rows can be inferred, and therefore, omitted

Exercise (E)	Φ
regular (r)	0.70
irregular (i)	0.30

Diet (D)	Φ
healthy (h)	0.25
unhealthy (u)	0.75

Heart disease (H)	E=r, D=h	E=r, D=u	E=i, D=h	E=i, D=u
yes (y)	0.25	0.40	0.55	0.80
no (n)	0.75	0.60	0.45	0.20

Blood pressure (B)	H=y	H=n
normal (l)	0.15	0.80
high (g)	0.85	0.20

Chest pain (C)	H=y	H=n
normal (m)	0.70	0.45
pain (p)	0.30	0.55

Classification using Bayesian networks

- Given no prior information, is a person suffering from heart disease?
- Essentially, a yes/no classification problem with some information
- Note that no other information (e.g., chest pain, etc.) are known
- Compute $P(H = y)$; if it is greater than $P(H = n)$, then predict "heart disease"

Classification using Bayesian networks

- Given no prior information, is a person suffering from heart disease?
- Essentially, a yes/no classification problem with some information
- Note that no other information (e.g., chest pain, etc.) are known
- Compute $P(H = y)$; if it is greater than $P(H = n)$, then predict "heart disease"

$$
\begin{aligned}
P(H = y) &= \sum_{\alpha,\beta} [P(H = y | E = \alpha, D = \beta).P(E = \alpha, D = \beta)] \\
&= \sum_{\alpha,\beta} [P(H = y | E = \alpha, D = \beta).P(E = \alpha).P(D = \beta)] \\
&= 0.25 \times 0.70 \times 0.25 + 0.40 \times 0.70 \times 0.75 \\
&\quad\quad + 0.55 \times 0.30 \times 0.25 + 0.80 \times 0.30 \times 0.75 \\
&= 0.475
\end{aligned}
$$

Classification using Bayesian networks (contd.)

- Given a person has high blood pressure, is she suffering from heart disease?
- Essentially, a yes/no classification problem with some information
- Note that not all information (e.g., chest pain, etc.) are known
- Compute $P(H = y|B = g)$; if it is greater than $P(H = n|B = g)$, then predict "heart disease"

Classification using Bayesian networks (contd.)

- Given a person has high blood pressure, is she suffering from heart disease?
- Essentially, a yes/no classification problem with some information
- Note that not all information (e.g., chest pain, etc.) are known
- Compute $P(H = y|B = g)$; if it is greater than $P(H = n|B = g)$, then predict "heart disease"

$$
\begin{aligned}
P(H = y|B = g) &= \frac{P(B = g|H = y).P(H = y)}{P(B = g)} \\
&= \frac{P(B = g|H = y).P(H = y)}{\sum_\alpha [P(B = g|H = \alpha).P(H = \alpha)]} \\
&= \frac{0.85 \times 0.475}{0.85 \times 0.475 + 0.20 \times 0.525} \\
&= 0.794
\end{aligned}
$$

Classification using Bayesian networks (contd.)

- Given a person has high blood pressure, unhealthy diet and irregular exercise, is she suffering from heart disease?
- Essentially, a yes/no classification problem with some information
- Note that not all information (e.g., chest pain, etc.) are known
- Compute $P(H = y|B = g, D = u, E = i)$; if it is greater than $P(H = n|B = g, D = u, E = i)$, then predict "heart disease"

Classification using Bayesian networks (contd.)

- Given a person has high blood pressure, unhealthy diet and irregular exercise, is she suffering from heart disease?
- Essentially, a yes/no classification problem with some information
- Note that not all information (e.g., chest pain, etc.) are known
- Compute $P(H = y|B = g, D = u, E = i)$; if it is greater than $P(H = n|B = g, D = u, E = i)$, then predict "heart disease"

$$
\begin{aligned}
& P(H = y|B = g, D = u, E = i) \\
& \quad = \frac{P(B = g|H = y, D = u, E = i).P(H = y|D = u, E = i)}{P(B = g|D = u, E = i)} \\
& \quad = \frac{P(B = g|H = y).P(H = y|D = u, E = i)}{\sum_\alpha [P(B = g|H = \alpha).P(H = \alpha|D = u, E = i)]} \\
& \quad = \frac{0.85 \times 0.80}{0.85 \times 0.80 + 0.20 \times 0.20} \\
& \quad = 0.944
\end{aligned}
$$

Training

- Two important steps

Training

- Two important steps
- First, learning the network topology
  - Which edges are present?

Training

- Two important steps
- First, learning the network topology
  - Which edges are present?
  - Domain knowledge from human experts

Training

- Two important steps
- First, learning the network topology
  - Which edges are present?
  - Domain knowledge from human experts
- Second, learning the CPTs

Training

- Two important steps
- First, learning the network topology
  - Which edges are present?
  - Domain knowledge from human experts
- Second, learning the CPTs
  - Same method as naïve Bayes
  - Empirical probabilities
  - If not categorical, use Gaussian

Discussion

- Models reality better

Discussion

- Models reality better
- Dependence or correlation does not indicate which is cause and which is effect
  - Class is boring and not paying attention

Discussion

- Models reality better
- Dependence or correlation does not indicate which is cause and which is effect
  - Class is boring and not paying attention
- Topology of network is very important

Discussion

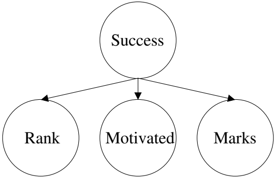
- Models reality better
- Dependence or correlation does not indicate which is cause and which is effect
  - Class is boring and not paying attention
- Topology of network is very important
- For large CPTs, require lots of training data

Discussion

- Models reality better
- Dependence or correlation does not indicate which is cause and which is effect
  - Class is boring and not paying attention
- Topology of network is very important
- For large CPTs, require lots of training data
- Naïve Bayes is a special case

Discussion

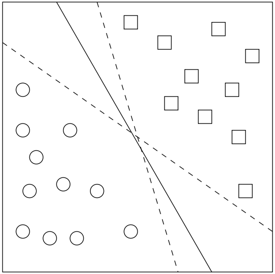
- Models reality better
- Dependence or correlation does not indicate which is cause and which is effect
    - Class is boring and not paying attention
- Topology of network is very important
- For large CPTs, require lots of training data
- Naïve Bayes is a special case
    - Class is parent and attributes are children



Outline

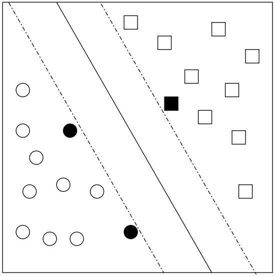
Support vector machines

- Support vector machine or SVM is a maximal margin classifier
- Binary classifier, i.e., two classes only
- It finds a hyperplane (called decision boundary) that separates the two classes
- Of multiple such hyperplanes, it finds the one whose distance or margin from the two classes is maximal
- Assumption is that classes are linearly separable



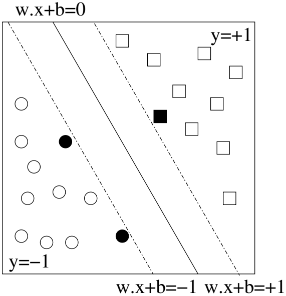
Support vectors

- Objects that are closest to the decision boundary on either side are called support vectors
- Optimal decision boundary and margin depend only on support vectors
- Support vectors are the most important objects
  - Optimal decision boundary will not change unless support vectors are changed
  - Other objects do not influence the decision boundary



Decision boundary

- Each object is represented as $\vec{x_i}$ with its corresponding class y_i
- For convenience, y_i is considered $+1$ or -1
- Decision boundary hyperplane is represented by $w.x + b = 0$
  - $\vec{w}$ essentially acts as weights on dimensions of $\vec{x}$
- *Support vectors* have $w.x + b = \pm 1$
  - w can always be scaled to achieve this
- For every object, $y_i(w.x_i + b) \geq 1$
  - Objects in class $y_i = +1$ have $w.x + b \geq +1$
  - Objects in class $y_i = -1$ have $w.x + b \leq -1$

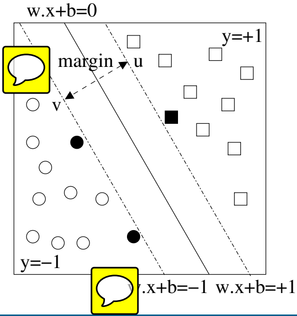


Margin

- Direction of $\vec{w}$ is perpendicular to decision boundary
- Margin is defined as the distance between the two hyperplanes $w.x + b = +1$ and $w.x + b = -1$
- Consider two points u and v on the two hyperplanes
- Margin d is distance between u and v

$$\vec{w}.(\vec{u} - \vec{v}) = 2$$

$$\therefore \; d = ||\vec{u} - \vec{v}|| = 2/||\vec{w}||$$



SVM problem specification

- SVM tries to maximize the margin d
- Constraints are on the objects
- Maximizing d is equivalent to minimizing $||w||$ or $||w||^2/2$

$$\min \frac{||w||^2}{2}$$
$$\text{s.t. } \forall i,\ y_i(w.x_i + b) \geq 1$$

- Convex (quadratic) optimization problem
- Lagrange multipliers λ_i for each object
- Karush-Kuhn-Tucker (KKT) conditions

SVM solution

- Essentially finds all λ_i and b
- Margin can then be expressed in terms of λ_i

$$\vec{w} = \sum_{\forall i} \lambda_i y_i \vec{x_i}$$

SVM solution

- Essentially finds all λ_i and b
- Margin can then be expressed in terms of λ_i

$$\vec{w} = \sum_{\forall i} \lambda_i y_i \vec{x_i}$$

- A test object x_q is classified by computing

$$sign(w.x_q + b) = sign\left(\sum_{\forall i} \lambda_i y_i \vec{x_i}.\vec{x_q} + b\right)$$

SVM solution

- Essentially finds all λ_i and b
- Margin can then be expressed in terms of λ_i

$$\vec{w} = \sum_{\forall i} \lambda_i y_i \vec{x_i}$$

- A test object x_q is classified by computing

$$sign(w.x_q + b) = sign\left(\sum_{\forall i} \lambda_i y_i \vec{x_i}.\vec{x_q} + b\right)$$

- Only for objects that are *support vectors*, $\lambda_i > 0$
- For all other objects, $\lambda_i = 0$
- Thus, complexity of testing is only the number of support vectors
- Complexity of training is enormous though

Dual problem specification

- Minimization problem can be converted to maximization by primal-dual transformation
- Dual formulation becomes

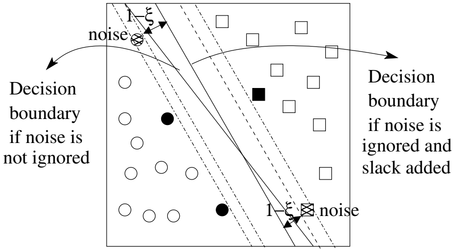
$$\max \sum_{\forall i} \lambda_i - \frac{1}{2} \sum_{\forall i} \sum_{\forall j} \lambda_i \lambda_j y_i y_j \vec{x_i} . \vec{x_j}$$

$$\text{s.t. } \forall i,\ \lambda_i \geq 0,\ \sum_{\forall i} \lambda_i y_i = 0$$

- Dual problem has only dot products of vectors

Handling noise

- SVM builds a classifier that is correct for *all* training objects
- If noise is present, decision boundary changes
- To handle noise, slack variables ξ_i are modeled
- For positive class, $w.x_i + b \geq +(1 - \xi_i)$
- For negative class, $w.x_i + b \leq -(1 - \xi_i)$
- Together, for every object, $y_i(w.x_i + b) \geq 1 - \xi_i$



SVM problem with slack

- Margin still remains $d = 2/||w||$
- It is called soft margin
- However, $||w||^2/2||$ cannot be simply minimized any more

SVM problem with slack

- Margin still remains $d = 2/||w||$
- It is called soft margin
- However, $||w||^2/2||$ cannot be simply minimized any more
- SVM may find a decision boundary with too many objects modeled as noise
- In other words, too much slack can be added

SVM problem with slack

- Margin still remains $d = 2/||w||$
- It is called soft margin
- However, $||w||^2/2||$ cannot be simply minimized any more
- SVM may find a decision boundary with too many objects modeled as noise
- In other words, too much slack can be added
- Hence, slack needs to be factored in the minimization as well

$$\min \frac{||w||^2}{2} + C.\sum_{\forall i} f(\xi_i)$$

$$\text{s.t. } \forall i,\ y_i(w.x_i + b) \geq 1 - \xi_i$$

- $f(\xi_i)$ is a monotonic function and can be simply ξ_i itself
- Solution yields Lagrange multipliers λ_i and slack variables ξ_i for each object

Non-linearly separable data

- Data may not be linearly separable
- Find a transformation ϕ from x space to $\phi(x)$ space
- Data *becomes* linearly separable in $\phi(x)$ space

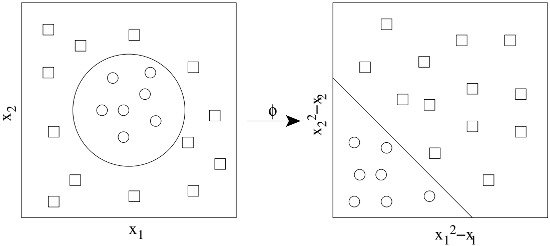
Example

- Suppose the decision boundary is a circle
- Centre is $0.5, 0.5$ and radius is 1
- Class is $+1$ if outside the circle, -1 otherwise
- Equation of decision boundary becomes

$$\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} = 1$$

$$\text{or, } x_1^2 - x_1 + x_2^2 - x_2 - 0.5 = 0$$

- In $(x_1^2 - x_1, x_2^2 - x_2)$ space, data becomes linearly separable



Transformation

- How to find such a transformation ϕ?

Transformation

- How to find such a transformation ϕ?
- Not obvious or easy at all
- Even if such a transformation exists, the transformed objects may reside in a very high-dimensional or *infinite* dimensional space
- How to use an SVM then?

Transformation

- How to find such a transformation ϕ?
- Not obvious or easy at all
- Even if such a transformation exists, the transformed objects may reside in a very high-dimensional or *infinite* dimensional space
- How to use an SVM then?
- Use the famous kernel trick
- A kernel is a function that computes the similarity between two vectors

Kernel trick

- Note that testing an object does not require value of w
- All it requires is an ability to compute *dot product* with the support vectors
- Same is true for training when dual of optimization problem is used
- Hence, testing can be simply written as

$$sign(w.\phi(x_q) + b) = sign\left(\sum_{\forall i} \lambda_i y_i \phi(\vec{x_i}).\phi(\vec{x_q}) + b\right)$$

- Use a kernel K that computes the dot product directly without transformation

$$K(x_i, x_j) = \phi(x_i).\phi(x_j)$$

Example of a kernel

- Vectors $\vec{u}$ and $\vec{v}$ are of dimensionality n
- Transformations $\phi(\vec{\cdot})$ are circles

$$\phi(\vec{u}) = \langle u_1 u_1, u_1 u_2, \ldots, u_n u_n, \sqrt{2} u_1, \ldots, \sqrt{2} u_n, 1 \rangle$$
$$\phi(\vec{v}) = \langle v_1 v_1, v_1 v_2, \ldots, v_n v_n, \sqrt{2} v_1, \ldots, \sqrt{2} v_n, 1 \rangle$$

Example of a kernel

- Vectors $\vec{u}$ and $\vec{v}$ are of dimensionality n
- Transformations $\phi(\vec{\cdot})$ are circles

$$\phi(\vec{u}) = \langle u_1 u_1, u_1 u_2, \ldots, u_n u_n, \sqrt{2}u_1, \ldots, \sqrt{2}u_n, 1 \rangle$$
$$\phi(\vec{v}) = \langle v_1 v_1, v_1 v_2, \ldots, v_n v_n, \sqrt{2}v_1, \ldots, \sqrt{2}v_n, 1 \rangle$$

$$\text{Dot product } \phi(\vec{u}).\phi(\vec{v}) = \sum_{i=1}^{n} \sum_{j=1}^{n} u_i u_j v_i v_j + \sum_{i=1}^{n} \sqrt{2}u_i \sqrt{2}v_i + 1$$
$$= (\vec{u}.\vec{v} + 1)^2$$

Example of a kernel

- Vectors $\vec{u}$ and $\vec{v}$ are of dimensionality n
- Transformations $\phi(\vec{\cdot})$ are circles

$$\phi(\vec{u}) = \langle u_1 u_1, u_1 u_2, \ldots, u_n u_n, \sqrt{2}u_1, \ldots, \sqrt{2}u_n, 1\rangle$$
$$\phi(\vec{v}) = \langle v_1 v_1, v_1 v_2, \ldots, v_n v_n, \sqrt{2}v_1, \ldots, \sqrt{2}v_n, 1\rangle$$

$$\begin{aligned}\text{Dot product } \phi(\vec{u}).\phi(\vec{v}) &= \sum_{i=1}^{n}\sum_{j=1}^{n} u_i u_j v_i v_j + \sum_{i=1}^{n} \sqrt{2}u_i \sqrt{2}v_i + 1 \\ &= (\vec{u}.\vec{v} + 1)^2\end{aligned}$$

$$\therefore K(\vec{u}, \vec{v}) = (\vec{u}.\vec{v} + 1)^2$$

- Dimensionality of $\phi(\cdot)$ is $n^2 + n + 1$
- Computation of kernel $K(\cdot, \cdot)$ requires only $O(n)$ computations

Example of kernels used

- Three kernels are most frequently used
- Polynomial kernel

$$K(u, v) = (u.v + 1)^h$$

- Gaussian radial basis kernel

$$K(u, v) = e^{-\frac{||u-v||^2}{2\sigma^2}}$$

- Sigmoid kernel

$$K(u, v) = tanh(\kappa u.v - \delta)$$

Extension to multiple classes

Extension to multiple classes

- One-against-one
  - Every class is compared against every other
  - For m classes, $m(m-1)/2 = O(m^2)$ classifiers
  - Majority voting to determine final class

Extension to multiple classes

- One-against-one
  - Every class is compared against every other
  - For m classes, $m(m-1)/2 = O(m^2)$ classifiers
  - Majority voting to determine final class
- One-against-others
  - For every class, belonging to class versus not in class
  - For m classes, m classifiers
  - Final class is one with highest value of $w.x + b$
  - Farthest away from margin

Discussion

- Some other better kernel may exist

Discussion

- Some other better kernel may exist
- For large training set, training time may be too impractical

Discussion

- Some other better kernel may exist
- For large training set, training time may be too impractical
- Not incremental

Discussion

- Some other better kernel may exist
- For large training set, training time may be too impractical
- Not incremental
- Suffers from class imbalance problem