CS685: Data Mining
Support Vector Machines

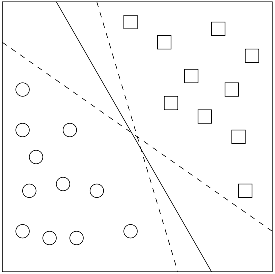
Arnab Bhattacharya
arnabb@cse.iitk.ac.in

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

1st semester, 2020-21
Mon 1030-1200 (online)

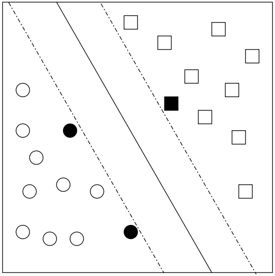
Support Vector Machines

- 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



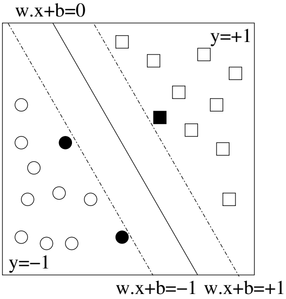
Support Vectors

- 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

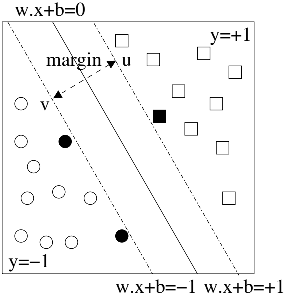
- Each object is represented as $\vec{x_i}$ with its corresponding class y_i
- For convenience, y_i is considered $+1$ or -1
- Decision boundary hyperplane is represented by $w.x + b = 0$
  - $\vec{w}$ essentially acts as weights on dimensions of $\vec{x}$
- *Support vectors* have $w.x + b = \pm 1$
  - 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

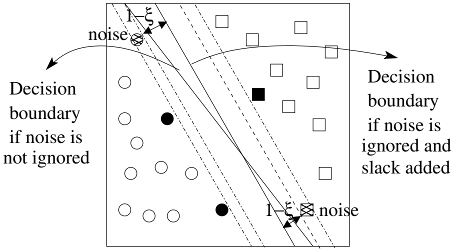
$$\max \sum_{\forall i} \lambda_i - \frac{1}{2} \sum_{\forall i} \sum_{\forall j} \lambda_i \lambda_j y_i y_j \vec{x_i}.\vec{x_j}$$

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

- 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

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

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

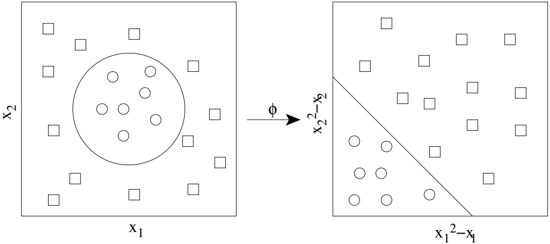
- $f(\xi_i)$ is a monotonic function and can be simply ξ_i itself
- Solution yields Lagrange multipliers λ_i and slack variables ξ_i for each object

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

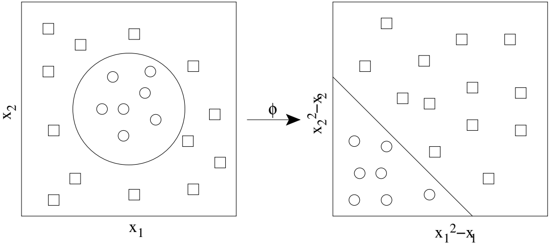
Example

- 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



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?
- 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

$$K(x_i, x_j) = \phi(\vec{x_i}).\phi(\vec{x_j})$$

Example of a Kernel

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

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

Example of a Kernel

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

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

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

Example of a Kernel

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

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

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

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

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

Popular Kernels

- 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

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

SVM for Multiple Classes

SVM for 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

SVM for 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

- For finite training set, transform that separates data linearly always exists
- Some other better kernel may exist

Discussion

- For finite training set, transform that separates data linearly always exists
- Some other better kernel may exist
- For large training set, training time may be too impractical

Discussion

- For finite training set, transform that separates data linearly always exists
- Some other better kernel may exist
- For large training set, training time may be too impractical
- Not incremental

Discussion

- For finite training set, transform that separates data linearly always exists
- Some other better kernel may exist
- For large training set, training time may be too impractical
- Not incremental
- Suffers from class imbalance problem

Discussion

- For finite training set, transform that separates data linearly always exists
- Some other better kernel may exist
- For large training set, training time may be too impractical
- Not incremental
- Suffers from class imbalance problem
- Non-linear kernels can overfit
- Slack guards against overfitting