Kernel Methods and Support Vector Machines

Oliver Schulte - CMPT 726

Bishop PRML Ch. 6

Support Vector Machines

Defining Characteristics

- Like logistic regression, good for continuous input features, discrete target variable.
- Like nearest neighbor, a *kernel method*: classification is based on weighted similar instances. The kernel defines similarity measure.
- Sparsity: Tries to find a few important instances, the *support vectors*.
- Intuition: Netflix recommendation system.

SVMs: Pros and Cons

Pros

- Very good classification performance, basically unbeatable.
- Fast and scaleable learning.
- Pretty fast inference.

Cons

- No model is built, therefore black-box.
- Not so applicable for discrete inputs.
- Still need to specify kernel function (like specifying basis functions).
- Issues with multiple classes, can use probabilistic version. (Relevance Vector Machine).

Two Views of SVMs

Theoretical View: linear separator

- SVM looks for linear separator but in *new feature space*.
- Uses a new criterion to choose a line separating classes: *max-margin*.

User View: kernel-based classification

- User specifies a kernel function.
- SVM learns weights for instances.
- Classification is performed by taking average of the labels of other instances, weighted by a) similarity b) instance weight.

Nice demo on web
`http://www.youtube.com/watch?v=3liCbRZPrZA`.

Two Views of SVMs

Theoretical View: linear separator

- SVM looks for linear separator but in *new feature space*.
- Uses a new criterion to choose a line separating classes: *max-margin*.

User View: kernel-based classification

- User specifies a kernel function.
- SVM learns weights for instances.
- Classification is performed by taking average of the labels of other instances, weighted by a) similarity b) instance weight.

Nice demo on web
`http://www.youtube.com/watch?v=3liCbRZPrZA`.

Two Views of SVMs

Theoretical View: linear separator

- SVM looks for linear separator but in *new feature space*.
- Uses a new criterion to choose a line separating classes: *max-margin*.

User View: kernel-based classification

- User specifies a kernel function.
- SVM learns weights for instances.
- Classification is performed by taking average of the labels of other instances, weighted by a) similarity b) instance weight.

Nice demo on web
http://www.youtube.com/watch?v=3liCbRZPrZA.

Two Views of SVMs

Theoretical View: linear separator

- SVM looks for linear separator but in *new feature space*.
- Uses a new criterion to choose a line separating classes: *max-margin*.

User View: kernel-based classification

- User specifies a kernel function.
- SVM learns weights for instances.
- Classification is performed by taking average of the labels of other instances, weighted by a) similarity b) instance weight.

Nice demo on web
`http://www.youtube.com/watch?v=3liCbRZPrZA`.

Example: X-OR

- X-OR problem: class of (x_1, x_2) is positive iff $x_1 \cdot x_2 > 0$.
- Use 6 basis functions
  $\phi(x_1, x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$.
- Simple classifier $y(x_1, x_2) = \phi_5(x_1, x_2) = \sqrt{2}x_1x_2$.
  - *Linear* in basis function space.
- Dot product $\phi(\boldsymbol{x})^T\phi(\boldsymbol{z}) = (1 + \boldsymbol{x}^T\boldsymbol{z})^2 = k(\boldsymbol{x}, \boldsymbol{z})$.
  - A quadratic kernel.

let's check SVM demo

http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml

Example: X-OR

- X-OR problem: class of (x_1, x_2) is positive iff $x_1 \cdot x_2 > 0$.
- Use 6 basis functions
  $\phi(x_1, x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$.
- Simple classifier $y(x_1, x_2) = \phi_5(x_1, x_2) = \sqrt{2}x_1x_2$.
  - *Linear* in basis function space.
- Dot product $\phi(\boldsymbol{x})^T\phi(\boldsymbol{z}) = (1 + \boldsymbol{x}^T\boldsymbol{z})^2 = k(\boldsymbol{x}, \boldsymbol{z})$.
  - A quadratic kernel.

let's check SVM demo

`http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml`

Valid Kernels

- Valid kernels: if $k(\cdot,\cdot)$ satisfies:
  - Symmetric; $k(\boldsymbol{x}_i,\boldsymbol{x}_j) = k(\boldsymbol{x}_j,\boldsymbol{x}_i)$
  - Positive definite; for any $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$, the Gram matrix $\boldsymbol{K}$ must be positive semi-definite:

$$\boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{x}_1,\boldsymbol{x}_1) & k(\boldsymbol{x}_1,\boldsymbol{x}_2) & \ldots & k(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N,\boldsymbol{x}_1) & k(\boldsymbol{x}_N,\boldsymbol{x}_2) & \ldots & k(\boldsymbol{x}_N,\boldsymbol{x}_N) \end{pmatrix}$$

  - Positive semi-definite means $\boldsymbol{x}^T\boldsymbol{K}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$

  then $k(\cdot,\cdot)$ corresponds to a dot product in some space ϕ

  - a.k.a. Mercer kernel, admissible kernel, reproducing kernel

Examples of Kernels

- Some kernels:
  - Linear kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^T \boldsymbol{x}_2$
  - Polynomial kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (1 + \boldsymbol{x}_1^T \boldsymbol{x}_2)^d$
    - Contains all polynomial terms up to degree d
  - Gaussian kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2 / 2\sigma^2)$
    - Infinite dimension feature space

Constructing Kernels

- Can build new valid kernels from existing valid ones:
  - $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = ck_1(\boldsymbol{x}_1, \boldsymbol{x}_2)$, $c > 0$
  - $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_1(\boldsymbol{x}_1, \boldsymbol{x}_2) + k_2(\boldsymbol{x}_1, \boldsymbol{x}_2)$
  - $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_1(\boldsymbol{x}_1, \boldsymbol{x}_2)k_2(\boldsymbol{x}_1, \boldsymbol{x}_2)$
  - $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(k_1(\boldsymbol{x}_1, \boldsymbol{x}_2))$
- Table on p. 296 gives many such rules

More Kernels

- Stationary kernels are only a function of the difference between arguments: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_1 - \boldsymbol{x}_2)$
  - Translation invariant in input space: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_1 + \boldsymbol{c}, \boldsymbol{x}_2 + \boldsymbol{c})$
- Homogeneous kernels, a. k. a. radial basis functions only a function of magnitude of difference: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(||\boldsymbol{x}_1 - \boldsymbol{x}_2||)$
- Set subsets $k(A_1, A_2) = 2^{|A_1 \cap A_2|}$, where $|A|$ denotes number of elements in A
- Domain-specific: think hard about your problem, figure out what it means to be similar, define as $k(\cdot, \cdot)$, prove positive definite.

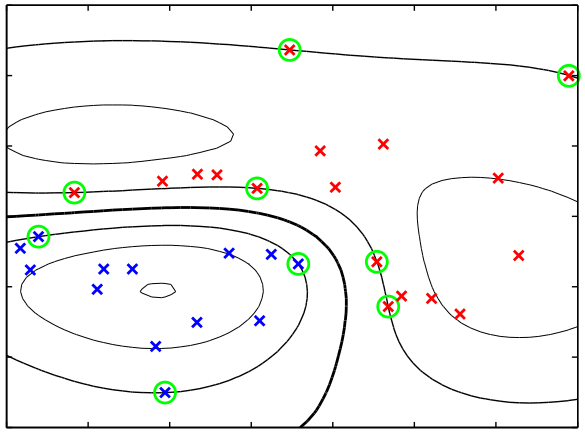
The Kernel Classification Formula

- Suppose we have a kernel function k and N labelled instances with weights $a_n \geq 0, n = 1, \ldots, N$.
- As with the perceptron, the target labels +1 are for positive class, -1 for negative class.
- Then

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b$$

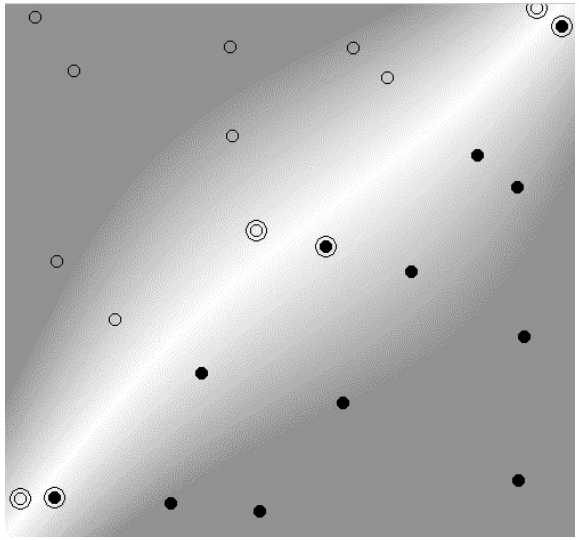
- $\boldsymbol{x}$ is classified as positive if $y(\boldsymbol{x}) > 0$, negative otherwise.
- If $a_n > 0$, then $\boldsymbol{x}_n$ is a support vector.
- Don't need to store other vectors.
- $\boldsymbol{a}$ will be sparse - many zeros.

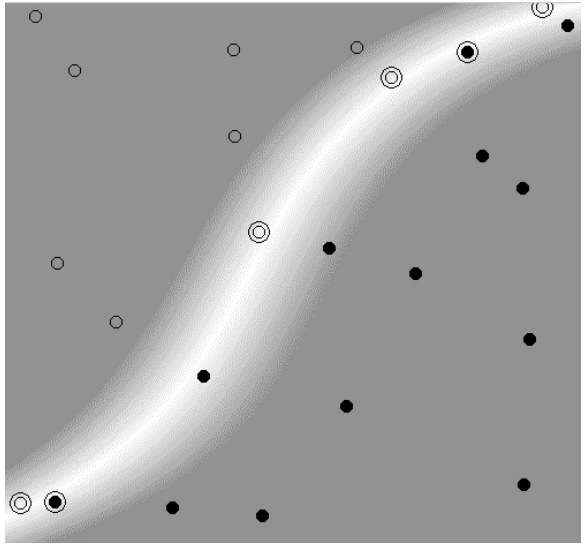
Examples



- SVM with Gaussian kernel
- Support vectors circled.
- They are the closest to the other class.
- Note non-linear decision boundary in x space

Examples





- From Burges, *A Tutorial on Support Vector Machines for Pattern Recognition* (1998)
- SVM trained using cubic polynomial kernel
  $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^T \boldsymbol{x}_2 + 1)^3$
- Left is linearly separable
  - Note decision boundary is almost linear, even using cubic polynomial kernel
- Right is not linearly separable
  - But is separable using polynomial kernel

Learning the Instance Weights

- The max-margin classifier is found by solving the following problem:
- Maximize wrt $\boldsymbol{a}$

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

  subject to the constraints
  - $a_n \geq 0, n = 1, \ldots, N$
  - $\sum_{n=1}^{N} a_n t_n = 0$

- It is quadratic, with linear constraints, convex in $\boldsymbol{a}$
- Bounded above since $\boldsymbol{K}$ positive semi-definite
- Optimal $\boldsymbol{a}$ can be found
  - With large datasets, descent strategies employed

Regression Kernelized

- Many classifiers can be written as using only dot products.
- Kernelization = replace dot products by kernel.
- E.g., the kernel solution for regularized least squares regression is

$$y(\boldsymbol{x}) = \boldsymbol{k}(\boldsymbol{x})^T(\boldsymbol{K} + \lambda \boldsymbol{I}_N)^{-1}\boldsymbol{t} \quad \text{vs.} \quad \boldsymbol{\phi}(\boldsymbol{x})(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda \boldsymbol{I}_M)^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}$$

  for original version
- N is number of datapoints (size of Gram matrix $\boldsymbol{K}$)
- M is number of basis functions (size of matrix $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$)
- Bad if $N > M$, but good otherwise
- $\boldsymbol{k}(\boldsymbol{x}) = (k(\boldsymbol{x}, \boldsymbol{x}_1, \ldots, k(\boldsymbol{x}, \boldsymbol{x}_n))$ is the vector of kernel values over data points x_n.

Conclusion

- Readings: Ch. 6.1-6.2 (pp. 291-297)
- Non-linear features, or domain-specific similarity measurements are useful
- Dot products of non-linear features, or similarity measurements, can be written as kernel functions
  - Validity by positive semi-definiteness of kernel function
- Can have algorithm work in non-linear feature space without actually mapping inputs to feature space
  - Advantageous when feature space is high-dimensional