Support Vector Machine Part 2

CS 760@UW-Madison
you should understand the following concepts

• the kernel trick
• polynomial kernel
• Gaussian/RBF kernel

• valid kernels and kernel algebra
• kernels and neural networks
Kernel Methods
Features

$\phi(x)$

Color Histogram

Extract features

$\mathbf{x}$

Red  Green
Features

Proper feature mapping can make non-linear to linear!

$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$(\frac{x_1}{a})^2 + (\frac{x_2}{b})^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$
Recall: SVM dual form

- Reduces to dual problem:
  \[
  \mathcal{L}(w, b, \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j
  \]

\[
\sum_i \alpha_i y_i = 0, \alpha_i \geq 0
\]

- Since \( w = \sum_i \alpha_i y_i x_i \), we have \( f(x) = w^T x + b = \sum_i \alpha_i y_i x_i^T x + b \)
Features

• Using SVM on the feature space \( \{\phi(x_i)\} \): only need \( \phi(x_i)^T \phi(x_j) \)

• Conclusion: no need to design \( \phi(\cdot) \), only need to design

\[
k(x_i, x_j) = \phi(x_i)^T \phi(x_j)
\]
Polynomial kernels

• Fix degree $d$ and constant $c$:
  $$k(x, x') = (x^T x' + c)^d$$

• What are $\phi(x)$?
• Expand the expression to get $\phi(x)$
Polynomial kernels

\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^2, \quad K(\mathbf{x}, \mathbf{x}') = (x_1 x'_1 + x_2 x'_2 + c)^2 = \begin{bmatrix} x_1^2 & \sqrt{2} x_1 x_2 \\ x_2^2 & \sqrt{2c} x_1 \\ \sqrt{2c} x_2 & \sqrt{2c} x'_1 \\ c & \sqrt{2c} x'_2 \end{bmatrix} \cdot \begin{bmatrix} x'_1^2 & \sqrt{2} x'_1 x'_2 \\ x'_2^2 & \sqrt{2c} x'_1 \\ \sqrt{2c} x'_2 & \sqrt{2c} x'_2 \\ c & \sqrt{2c} x'_2 \end{bmatrix}

Figure from Foundations of Machine Learning, by M. Mohri, A. Rostamizadeh, and A. Talwalkar
SVMs with polynomial kernels

Figure from Ben-Hur & Weston,
*Methods in Molecular Biology* 2010
Gaussian/RBF kernels

- Fix bandwidth $\sigma$:
  $$k(x, x') = \exp\left(-\|x - x'\|^2 / 2\sigma^2\right)$$
- Also called radial basis function (RBF) kernels

- What are $\phi(x)$? Consider the un-normalized version
  $$k'(x, x') = \exp(x^T x'/\sigma^2)$$
- Power series expansion:
  $$k'(x, x') = \sum_{i}^{+\infty} \frac{(x^T x')^i}{\sigma^i i!}$$
The RBF kernel illustrated

\[ k(x, x') = \exp(-\gamma \|x - x'\|^2) \]

Figures from openclassroom.stanford.edu (Andrew Ng)
Polynomial and RBF kernels

- Polynomial kernel with degree $d$ and constant $c$:
  \[ k(x, x') = (x^T x' + c)^d \]

- RBF kernels with bandwidth $\sigma$:
  \[ k(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \]
Mercer’s condition for kernels

- Theorem: $k(x, x')$ has expansion
  
  $k(x, x') = \sum_{i}^{+\infty} a_i \phi_i(x)\phi_i(x')$

  for nonnegative $a_i$’s, if and only if for any function $c(x)$,

  $\int \int c(x)c(x')k(x, x')dx dx' \geq 0$

(Omit some math conditions for $k$ and $c$)
Constructing new kernels

- Kernels are closed under positive scaling, sum, product, pointwise limit, and composition with a power series
  \[ \sum_{i}^{+\infty} a_i k^i(x, x') \]

- Example: \( k_1(x, x'), k_2(x, x') \) are kernels, then also is
  \[ k(x, x') = 2k_1(x, x') + 3k_2(x, x') \]

- Example: \( k_1(x, x') \) is kernel, then also is
  \[ k(x, x') = \exp(k_1(x, x')) \]
Kernel algebra

- given a valid kernel, we can make new valid kernels using a variety of operators

<table>
<thead>
<tr>
<th>kernel composition</th>
<th>mapping composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k(x, v) = k_a(x, v) + k_b(x, v)$</td>
<td>$\phi(x) = (\phi_a(x), \phi_b(x))$</td>
</tr>
<tr>
<td>$k(x, v) = \gamma k_a(x, v), \gamma &gt; 0$</td>
<td>$\phi(x) = \sqrt{\gamma} \phi_a(x)$</td>
</tr>
<tr>
<td>$k(x, v) = k_a(x, v)k_b(x, v)$</td>
<td>$\phi_i(x) = \phi_{ai}(x)\phi_{bi}(x)$</td>
</tr>
<tr>
<td>$k(x, v) = x^T A v, A$ is p.s.d.</td>
<td>$\phi(x) = L^T x, \text{ where } A = LL^T$</td>
</tr>
<tr>
<td>$k(x, v) = f(x)f(v)k_a(x, v)$</td>
<td>$\phi(x) = f(x)\phi_a(x)$</td>
</tr>
</tbody>
</table>
Kernels v.s. Neural Networks
Features

\[ x \]

Extract features

Color Histogram

\[ y = w^T \phi(x) \]

\[ \text{Red} \quad \text{Green} \]
Features: part of the model

Nonlinear model

Linear model

$$y = w^T \phi(x)$$

build hypothesis
Polynomial kernels

$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^2, \quad K(\mathbf{x}, \mathbf{x}') = (x_1 x'_1 + x_2 x'_2 + c)^2 = \begin{bmatrix} x_1^2 & \sqrt{2} x_1 x_2 \\ x_2^2 & \sqrt{2} c x_1 \\ \sqrt{2} x_1 x_2 & \sqrt{2} c x_1' \\ \sqrt{2} c x_2 & \sqrt{2} c x_2' \\ c & c \end{bmatrix} \cdot \begin{bmatrix} x'_1^2 & \sqrt{2} x'_1 x'_2 \\ x'_2^2 & \sqrt{2} c x'_1 \\ \sqrt{2} x'_1 x'_2 & \sqrt{2} c x'_1' \\ \sqrt{2} c x'_2 & \sqrt{2} c x'_2' \\ c & c \end{bmatrix}$

Figure from Foundations of Machine Learning, by M. Mohri, A. Rostamizadeh, and A. Talwalkar
Polynomial kernel SVM as two layer neural network

First layer is fixed. If also learn first layer, it becomes two layer neural network

\[
y = \text{sign}(w^T \phi(x) + b)
\]
Comments on SVMs

• we can find solutions that are globally optimal (maximize the margin)
  • because the learning task is framed as a convex optimization problem
  • no local minima, in contrast to multi-layer neural nets

• there are two formulations of the optimization: primal and dual
  • dual represents classifier decision in terms of support vectors
  • dual enables the use of kernel functions

• we can use a wide range of optimization methods to learn SVM
  • standard quadratic programming solvers
  • SMO [Platt, 1999]
  • linear programming solvers for some formulations
  • etc.
• kernels provide a powerful way to
  • allow nonlinear decision boundaries
  • represent/compare complex objects such as strings and trees
  • incorporate domain knowledge into the learning task
• using the kernel trick, we can implicitly use high-dimensional mappings without explicitly computing them
• one SVM can represent only a binary classification task; multi-class problems handled using multiple SVMs and some encoding
• empirically, SVMs have shown (close to) state-of-the-art accuracy for many tasks
• the kernel idea can be extended to other tasks (anomaly detection, regression, etc.)
THANK YOU

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, and Pedro Domingos.