Nonlinear Classifiers II

Nonlinear Classifiers: Introduction

- Classifiers
  - Supervised Classifiers
    - Linear Classifiers
    - Perceptron
    - Least Squares Methods
    - Linear Support Vector Machine
  - **Nonlinear Classifiers**
    - Part I: Multi Layer Neural Networks
    - **Part II: Polynomial Classifier, RBF, Nonlinear SVM**
  - Decision Trees
  - Unsupervised Classifiers
Nonlinear Classifiers: Agenda

Part II: Nonlinear Classifiers

- **Polynomial Classifier**
  - Special case of a Two-Layer Perceptron
  - Activation function with non linear input

- **Radial Basis Function Network**
  - Special case of a two-layer network
  - Radial Basis activation Function
  - Training is simpler and faster

- **Nonlinear Support Vector Machine**

Polynomial Classifier: XOR problem

- XOR problem with polynomial function.
  - With nonlinear polynomial function classes can be classified.
  - Example XOR-Problem:

![Diagram of XOR problem](image-url)
Polynomial Classifier: XOR problem

• XOR problem with polynomial function.
  • With nonlinear polynomial functions, classes can be classified.
  • Example XOR-Problem:

\[ \phi : X \rightarrow H \]

\[ z = \phi(x) \]

...but with a polynomial function!

Polynomial Classifier: XOR problem

With \( z = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix} \) we obtain:

\[ \phi(0,0) \rightarrow (0,0,0) \]
\[ \phi(0,1) \rightarrow (0,1,0) \]
\[ \phi(1,0) \rightarrow (1,0,0) \]
\[ \phi(1,1) \rightarrow (1,1,1) \]

... that's separable in \( H \) by the Hyperplane:

\[ g(z) = \frac{1}{4} - 1z_1 - 1z_2 + 2z_3 = 0 \]
Polynomial Classifier: XOR problem

Hyperplane: \( g(y) = w \cdot y + w_0 = 0 \)

\[ g(z) = \frac{1}{4} - z_1 - z_2 + 2z_3 = 0 \]

is Hyperplane in \( H \)

\[ g(x) = \frac{1}{4} - x_1 - x_2 + 2x_1x_2 \]

is Polynom in \( X \)

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( z_1 )</th>
<th>( z_2 )</th>
<th>( z_3 )</th>
<th>( \omega )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A (true)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>B (false)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>B (false)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A (true)</td>
</tr>
</tbody>
</table>

Decision Surface in \( X \)

\[ g(z) = \frac{1}{4} - 1x_1 - 1x_2 + 2x_1x_2 \geq 0 \quad x \in A \]

\[ < 0 \quad x \in B \]

\[ x_z = (x_1 - 0.25)/(2x_1 - 1) \]

MatLab:

```matlab
>> x1=[-0.5:0.1:1.5];
>> x2=(x1-0.25)/(2*x1-1);
>> plot(x1,x2);
```
Polynomial Classifier: XOR problem

- With nonlinear polynomial functions, classes can be classified in original space $X$.
  - Example: XOR-Problem

$$ z = \phi(x) $$

was not linear separable!
... but linear separable in $H$!
... and separable in $X$ with a polynomial function!

Polynomial Classifier

more general

- Decision function is approximated by a polynomial function $g(x)$, of order $p$ e.g. $p = 2$:

$$ g(x) = w_0 + \sum_{i=1}^{l} w_i x_i + \sum_{i=1}^{l} \sum_{m=1}^{l-1} w_{mn} x_i x_m + \sum_{i=1}^{l} w_{mi} x_i^2 $$

$$ g(x) = w^T \bar{z} + w_0, $$

with

$$ w^T = [w_1, w_2, w_{12}, w_{11}, w_{22}], $$

$$ \bar{z} = [x_1, x_2, x_1 x_2, x_1^2, x_2^2]^T \text{ and } x = [x_1, x_2]^T $$

- Special case of a Two-Layer Perceptron
- Activation function with polynomial input
Nonlinear Classifiers: Agenda

Part II: Nonlinear Classifiers

- Polynomial Classifier
- **Radial Basis Function Network**
  - Special case of a two-layer network
  - Radial Basis activation Function
  - Training is simpler and faster
- Nonlinear Support Vector Machine
- Application: ZIP Code, OCR, FD (W-RVM)
- Demo: libSVM, DHS or Hlavac

Radial Basis Function

- Radial Basis Function Networks (RBF)
  - Choose \( g(x) = w_0 + \sum_{i=1}^{k} w_i g_i(x) \)
    with \( g_i(x) = \exp \left( \frac{-\|x - \hat{c}_i\|^2}{2\sigma_i^2} \right) \)
Radial Basis Function

\[ g(x) = w_0 + \sum_{i=1}^{k} w_i g_i(x) \]

with \( g_i(x) = \exp \left( -\frac{(x - c_i)^2}{2\sigma_i^2} \right) \)

Examples:

\[
\begin{align*}
&c_i = 2.5, 0.0, 1.0, 1.5, 2.0, \\
&t = 1, \ldots, k, \\
&k = 5, \\
&\sigma = 1/\sqrt{2}
\end{align*}
\]

How to choose \( c, \sigma, k \)?

Radial Basis Function

- Radial Basis Function Networks (RBF)
  - Equivalent to a single layer network, with RBF activations and linear output node.
Radial Basis Function: XOR problem

\[ \xi = \phi(x) \]

\[ g(x) = z_1 + z_2 - 1 = 0 \]
\[ g(x) = \exp(-\|x - c_1\|^2) + \exp(-\|x - c_2\|^2) - 1 = 0 \]

... not linear separable pattern set in \( X \).
... separable using a nonlinear function (RBF) in \( X \) that separates the set in \( H \) with a linear decision hyperplane!

Radial Basis Function

- Decision function as summation of \( k \) RBF’s

\[ g(x) = w_0 + \sum_{i=1}^{k} w_i \exp\left(-\frac{(x - c_i)^T(x - c_i)}{2\sigma_i^2}\right) \]

- Training of the RBF networks
  1. Fixed centers: Choose centers randomly among the data points. Also fix \( \sigma_i \)'s. Then \( g(x) = w_0 + w^T \xi \) is a typical linear classifier design.
  2. Training of the centers: This is a nonlinear optimization task.
  3. Combine supervised and unsupervised learning procedures.
  4. The unsupervised part reveals clustering tendencies of the data and assigns the centers at the cluster representatives.
Nonlinear Classifiers: Agenda

Part II: Nonlinear Classifier

- Polynomial Classifier
- Radial Basis Function Network
- **Nonlinear Support Vector Machine**
  - Application: ZIP Code, OCR, FD (W-RVM)
  - Demo: libSVM, DHS or Hlavac

Nonlinear Classifiers: SVM

XOR problem:

- linear separation in high dimensional space $H$ via nonlinear functions (polynomial and RBF’s) in the original space $X$.
- for this we found nonlinear mappings $\phi(x): X \rightarrow H$

Is that possible without knowing the mapping function $\phi$ ?!?
Non-linear Support Vector Machines

- Recall that, the probability of having linearly separable classes increases as the dimensionality of feature vectors increases.

Assume the mapping:

$$x \in \mathbb{R}^l \rightarrow z \in \mathbb{R}^k, \quad k > l$$

\[\rightarrow\] Then use linear SVM in \(\mathbb{R}^k\)

Non-linear SVM

- **Support Vector Machines:** with \(x \rightarrow z \in \mathbb{R}^k\)

- Recall that in this case the dual problem formulation will be

$$\max_{\lambda} \left( \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j z_i^T z_j \right)$$

where \(z_i \in \mathbb{R}^k\), \(y \in \{-1, 1\}\) (class labels)

- the classifier will be

$$g(z) = w^T z + w_0$$

$$= \sum_{i=1}^{N} \lambda_i y_i z_i^T z + w_0$$
Non-linear SVM

- Thus, only inner products in a high dimensional space are needed!

=> Something clever (kernel trick):
Compute the inner products in the high dimensional space as functions of inner products performed in the low dimensional space!!!
Non-linear SVM

- Mercer’s Theorem

Let \( x \rightarrow \phi(x) \in H \)

To guarantee that the symmetric function \( K(x_i, x_j) \) (kernel) can be represented as

\[
\sum_r \phi_r(x_i)\phi_r(x_j) = K(x_i, x_j)
\]

that is an inner product in \( H \),

it is necessary and sufficient that

\[
\int K(x_i, x_j) g(x_i)g(x_j) \, dx_i dx_j \geq 0 \quad (1)
\]

for any \( g(x) \):

\[
\int g^2(x) \, dx < +\infty \quad (2)
\]

Non-linear SVM

- Kernel Function

- So, any kernel \( K(x, y) \) satisfying (1) & (2), corresponds to an inner product in SOME space!!!

- Kernel trick: We do not have to know the mapping function, but for some kernel functions we try to linearly separate pattern sets in a high dimensional space only using a function of the inner product in the original space.
Non-linear SVM

- **Kernel Functions:** Examples
  - Polynomial: \( K(x_i, x_j) = (x_i^T x_j + 1)^q, \quad q > 0 \)
  - Radial Basis Functions:
    \[
    K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{\sigma^2}\right)
    \]
  - Hyperbolic Tangent:
    \[
    K(x_i, x_j) = \tanh(\beta x_i^T x_j + \gamma)
    \]
    for appropriate values of \( \beta, \gamma \)
    (e.g. \( \beta = 2 \) and \( \gamma = 1 \)).

**Support Vector Machines Formulation**

- Step 1: Choose appropriate kernel. This implicitly assumes a mapping to a higher dimensional (yet, not known) space.
Non-linear SVM

SVM Formulation

• Step 2:

\[ \hat{\lambda} = \arg \max_\lambda \left( \sum_{i} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j K(x_i, x_j) \right) \]

subject to: \[ 0 \leq \lambda_i \leq C, \quad i = 1, 2, \ldots, N \]
\[ \sum_i \lambda_i y_i = 0 \]

This results to an implicit combination

\[ w = \sum_{i=1}^{N} \lambda_i y_i \Phi(x_i) \]

Non-linear SVM

- SVM Formulation

• Step 3: Assign \( x \) to

\[ \omega_1 \text{ if } g(x) = \sum_{i=1}^{N} \lambda_i y_i K(x_i, x) + w_0 \geq 0 \]
\[ \omega_2 \text{ if } g(x) = \sum_{i=1}^{N} \lambda_i y_i K(x_i, x) + w_0 < 0 \]
Non-linear SVM

- **SVM: The non-linear case**
  - The SVM Architecture
    - SVM special case of a two-layer neural network with special activation function and a different learning method.
    - Their attractiveness comes from their good generalization properties and simple learning.

![Diagram of SVM architecture](image)

Non-linear SVM

- **Linear SVM – Pol. SVM** in the input space $X$

![Graph showing linear SVM and polynomial SVM](image)

Training Error: 0.276
Test Error: 0.268
Bayes Error: 0.210
Non-linear SVM

- Pol. SVM – RBF SVM in the input space $X$

Nonlinear Classifiers: SVM

- Pol. SVM – RBF SVM in the input space $X$
Nonlinear Classifiers: SVM

- **Software**

  - **SVMlight**: Thorsten Joachims - free software in C, known for quality and speed.

  - **LIB SVM**: free software based on Platt’s SMO algorithm and Joachims code, written by Chih-Chung Chang and Chih-Jen Lin.

  - **Equiols**: Commercial software package which automates the tuning and model selection with SVMs