Predictive Modelling

Predicting a variable of interest $y$ given knowledge of some other variables $x$
- Classification: $y$ is categorical
- Regression: $y$ is real-valued

Overview
- Classification, Regression models
- SVMs
- Inductive Logic Programming

Reading: HMS chapters 10,11

Classification

How should we assign example $x$ to a class $C_k$?
1. use discriminant functions $y_k(x)$
2. model class-conditional densities $P(x|C_k)$ and then use Bayes’ rule
3. Model posterior probabilities $P(C_k|x)$ directly

Classification methods

- Perceptron (1)
- Logistic regression (3)
- Fisher’s Linear Discriminant (1)
- Decision Trees (1)
- Nearest Neighbour (1)
- Naive Bayes (2)
- Class-conditional modelling with Gaussians/Gaussian mixtures (2)
- Neural Networks (MLPs/RBFS) (3)
- Kernel machines (e.g. SVMs (1))
- Inductive Logic Programming (1)
**Decision Trees**

```
<table>
<thead>
<tr>
<th>y</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```


**Choosing which feature to split on**

Base on "purity" of instances at a leaf

```
<table>
<thead>
<tr>
<th></th>
<th>x1 = 0</th>
<th>x1 = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00/400</td>
<td>100/400</td>
<td></td>
</tr>
<tr>
<td>190/400</td>
<td>210/400</td>
<td></td>
</tr>
</tbody>
</table>
```

- Entropy / Information gain
  
  \[-(p_+ \log p_+ + p_- \log p_-)\]

- Multiple classes – predict most frequent at leaf
- Regression – predict average at leaf, and use variance for splitting

**Decision Trees Pros/Cons**

**Pro:**
- Easy to interpret
- Irrelevant attributes ignored
- Can handle missing data
- Compact (\#nodes much less than D after pruning)
- Greedy: fast to train. Also very fast to test

**Con:**
- Can’t split on second-order features
- Greedy: might not be best tree → unstable
- Accuracy often not great.

**Logistic Regression**

- Linear function of attributes, passed through logistic "squashing" function. Decision boundary 0.5.
- Learn parameters via gradient descent on negative log likelihood.
- For multiclass – use the softmax function (relative exponentials).
The power of non-linear basis functions

Using two Gaussian basis functions $\phi_1(x)$ and $\phi_2(x)$

Figure credit: Chris Bishop, PRML

We can transform the input space if we want $x \rightarrow \phi(x)$

Regression methods

include

- Linear parameter models
- Nearest neighbour
- Decision trees
- Neural Networks (MLPs/RBFS)
- Kernel machines (e.g. SVR)

When To Consider Nearest Neighbour

- Instances map to points in $\mathbb{R}^n$
- Less than 20 attributes per instance
- Lots of training data

Advantages:

- Training is very fast
- Learn complex target functions
- Don’t lose information

Disadvantages:

- Slow at query time
- Easily fooled by irrelevant attributes

When to Consider Neural Networks

- Input is high-dimensional discrete or real-valued (e.g. raw sensor input)
- Output is discrete or real valued
- Possibly noisy data
- Form of target function is unknown
- Human readability of result is unimportant
- Output can be a vector of values
- Can afford long training times (but fast at runtime)

Acknowledgement: This slide and the previous one are based on slides produced by Tom Mitchell, available from http://www.cs.cmu.edu/~tom/
Support Vector Machines

Overview
- Supervised Learning Problem
- Two key ideas for SVMs
  - Learn separating hyperplane with maximum margin
  - Expand input into a high-dimensional space
- Very widely used and successful
- Reading: SVM Handout

Separating Hyperplane
- Training instances \((x_i, y_i), i = 1, \ldots, n\). \(y_i \in \{-1, +1\}\)
- Hyperplane \(w \cdot x + w_0 = 0\)

Maximum margin
- Let the perpendicular distance from the hyperplane to the nearest +1 class point be \(d_+\)
- Similarly for nearest class -1 point, perpendicular distance is \(d_-\)
- Margin is defined as \(\min(d_+, d_-)\)
- Support vector machine algorithm looks for \((w, w_0)\) that gives rise to the maximum margin
- Choose hyperplane so that \(d_+ = d_-\)
- Note that \((w, w_0)\) and \((cw, cw_0)\) defines the same hyperplane.
- Remove rescaling freedom by demanding that \(\min_{x_i} |w \cdot x_i + w_0| = 1\)
Maximum margin

Now we want the training data to satisfy the constraints

\[ w \cdot x_i + w_0 \geq +1 \quad \text{for } y_i = +1 \]
\[ w \cdot x_i + w_0 \leq -1 \quad \text{for } y_i = -1 \]

These can be combined as

\[ y_i (w \cdot x_i + w_0) - 1 \geq 0 \quad \text{for all } i \]

Margin has size \( \frac{1}{||w||} \)

So to maximize the margin we can minimize \( ||w||^2 \) subject to the constraints

\[ y_i (w \cdot x_i + w_0) - 1 \geq 0 \quad \text{for all } i \]

Non-separable training sets

Add a “slack” variable \( \xi_i \geq 0 \) for each training example

New optimization problem is to minimize

\[ ||w||^2 + C (\sum_{i=1}^{n} \xi_i)^k \]

subject to the constraints

\[ w \cdot x_i + w_0 \geq 1 - \xi_i \quad \text{for } y_i = +1 \]
\[ w \cdot x_i + w_0 \leq -1 + \xi_i \quad \text{for } y_i = -1 \]

Usually set \( k = 1 \). \( C \) is a trade-off parameter, picked by hand (see below). Large \( C \) gives a large penalty to errors

Optimal hyperplane

Optimal hyperplane can be computed from a quadratic programming problem using Lagrange multipliers

\[ w = \sum_i \alpha_i y_i x_i \]

Optimal hyperplane is determined by just a few examples: call these support vectors

\( \alpha_i = 0 \) for non-support patterns

Optimization problem has no local minima (cf MLPs)

Prediction on new data point \( x \)

\[ f(x) = \text{sgn}(w \cdot x + w_0) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i (x_i \cdot x) + w_0\right) \]
Non-linear SVMs

- Transform \( x \) to \( \tilde{x} \) in some other space using the mapping \( \Phi \)
  \[ \tilde{x} = \Phi(x) \]

- Linear algorithm depends only on \( x \cdot x_i \). Hence transformed algorithm depends only on \( \Phi(x) \cdot \Phi(x_i) \)

- Use a kernel function \( k(x_i, x_j) \) such that
  \[ k(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \]

- Example 1 for 2-d input space
  \[ \Phi(x) = \begin{pmatrix} x_1^2 & \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \]
  \[ k(x_i, x_j) = (x_i \cdot x_j)^2 \]

A non-linear SVM decision boundary obtained using the kernel \( k(x, y) = (x \cdot y)^3 \)

Choosing \( \Phi, C \)

- There are VC dimension bounds that relate to SVM machines
- The Structural Risk Minimization principle seeks a minimum of the sum of two terms involving empirical error and the VC-dimension of the hypothesis class
- Alternatively, cross-validation methods can be used

Example 2
\[ k(x_i, x_j) = \exp \left( -\frac{||x_i - x_j||^2}{2\sigma^2} \right) \]
In this case the dimension of \( \Phi \) is infinite

To test a new input \( x \)
\[ f(x) = \text{sgn} \left( \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b \right) \]
Applications

\[ \sum f(x) = \text{sgn} \left( \sum \lambda_i k(x, x_i) + b \right) \]

input vector \( x \)
support vectors \( x_1, \ldots, x_4 \)

comparison:

\[ k(x, x_i) = \exp\left(-\frac{||x - x_i||^2}{c}\right) \]

\[ k(x, x_i) = \tanh(\kappa (x . x_i) + \theta) \]

\[ k(x, x_i) = (x . x_i)^d \]

Examples

US Postal Service digit data (7291 examples, \( 16 \times 16 \) images). Three SVMs using polynomial, RBF and MLP-type kernels were used.

Use almost the same (\( \approx 90\% \)) small sets (4% of data base) of SVs.

All systems perform well (\( \approx 4\% \) error).

Many other applications, e.g.

- Text categorization
- Face detection
- DNA analysis

Support Vector Regression

- The support vector algorithm can also be used for regression problems.
- Instead of using squared-error, the algorithm uses the \( \epsilon \)-insensitive error:

\[ E_\epsilon(z) = \begin{cases} 
|z| - \epsilon & \text{if } |z| \geq \epsilon, \\
0 & \text{otherwise}. 
\end{cases} \]

- Again a sparse solution is obtained from a QP problem:

\[ y(x) = \sum_{i=1}^{n} \beta_i k(x, x_i) + w_0 \]

The data points within the \( \epsilon \)-insensitive region have \( \beta_i = 0 \).
SVM summary

- Learn linear decision boundaries (cf. perceptrons)
- Pick hyperplane that maximizes margin
- Use slack variables to deal with non-separable data
- Optimal hyperplane can be written in terms of support patterns
- Transform to higher-dimensional space using kernel functions
- Good empirical results on many problems
- Appears to avoid overfitting in high dimensional spaces (cf. regularization)

Further SVM resources

- Kernel machines website http://www.kernel-machines.org/

Inductive Logic Programming

- Learning First-Order Rules: Going beyond an attribute-based representation
- Consider learning the concept of Daughter from data concerning pairs of people, e.g.
  \[(Name_1 = Sharon, Mother_1 = Louise, Father_1 = Bob, Male_1 = False, Female_1 = True, Name_2 = Bob, Mother_2 = Nora, Father_2 = Victor, Male_2 = True, Female_2 = False, Daughter_{1,2} = True)\]
- Giving this information to a propositional rule learner (e.g. C4.5) would give rise to rules such as
  \[IF (Father_1 = Bob) \land (Name_2 = Bob) \land (Female_1 = True) THEN Daughter_{1,2} = True\]
- Correct, but highly specific

- Need to describe relations among the values of the attributes
- Using first-order representations we could learn the rule
  \[IF Father(y, x) \land Female(y) THEN Daughter(x, y)\]
  where \(x\) and \(y\) are variables that can be bound to any person
- Another example GrandDaughter
  \[IF Father(y, z) \land Mother(z, x) \land Female(y) THEN GrandDaughter(x, y)\]
  Note existential quantification over \(z\)
Horn Clauses, PROLOG

- Horn clause
  \[ H \leftarrow (L_1 \land \ldots \land L_k) \]
  or equivalently
  \[ \text{IF } (L_1 \land \ldots \land L_k) \text{ THEN } H \]
- PROLOG programs are expressed as collections of Horn clauses
- ILP can be thought of as learning PROLOG programs

Approaches to ILP

- Top-down learning methods (e.g. FOIL; Quinlan 1990)
  - Sequential covering algorithms
  - General idea: start with a very general rule and specialize it to fit the data
- Induction as Inverted Deduction
  - General idea: start from the observations, work backwards

Examples of ILP in action

- Learning what chemical substructures produce mutagenic activity (Srinivasan et al, 1994)
- Learning a recursive definition of QUICKSORT
- Learning to design Finite Element meshes

Summary of ILP

- Need for hypotheses based on relational descriptions
- Power of first-order logic cf propositional logic
- Complexity of search sequential covering algorithms
- Dealing with uncertainty ⇒ stochastic ILP (Muggleton)
- Further reading: Mitchell, chapter 10

Acknowledgement: These slides on ILP are based on slides produced by Tom Mitchell.