Classification with microarray data

Aron Charles Eklund
eklund@cbs.dtu.dk

January 9, 2009
What to expect from Aron today

What is classification, and why

How to do classification with microarray data

After lunch: case study.

Then, classification exercises.
What is classification?

Given an object with a set of features (input data), a classifier assigns the objects to a class.

<table>
<thead>
<tr>
<th>features</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>34</td>
</tr>
<tr>
<td>sex</td>
<td>male</td>
</tr>
<tr>
<td>income</td>
<td>$$</td>
</tr>
</tbody>
</table>

A classifier is simply a set of rules or an algorithm

classifier algorithm:

if (age > 40) AND (income > $$)
then vote = McCain
else vote = Obama
Why is classification useful?

1. Classification as an end in itself
   – diagnosis of disease
   – predict response to chemotherapy
   – identify bacterial strains

2. Learn from the classifier
   – identify drug targets

classifier algorithm:

if (age > 40) AND (income > $$)
    then vote = republican
else vote = democrat
The challenge

Given a *training set* (data of known class), design a classifier that **accurately** predicts the class of novel data.

<table>
<thead>
<tr>
<th>x01</th>
<th>x02</th>
<th>x03</th>
<th>x04</th>
<th>x05</th>
<th>x06</th>
<th>x07</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>43</td>
<td>65</td>
<td>34</td>
<td>22</td>
<td>44</td>
<td>29</td>
</tr>
<tr>
<td>sex</td>
<td>female</td>
<td>male</td>
<td>male</td>
<td>female</td>
<td>female</td>
<td>male</td>
</tr>
<tr>
<td>income</td>
<td>$$$$$</td>
<td>$$$$$</td>
<td>$$$$$$</td>
<td>$</td>
<td>$$$$$$</td>
<td>$</td>
</tr>
<tr>
<td>vote</td>
<td>Obama</td>
<td>McCain</td>
<td>Obama</td>
<td>McCain</td>
<td>Obama</td>
<td>McCain</td>
</tr>
</tbody>
</table>

This is also called *statistical learning* or *machine learning*.
Classification as mapping

A classifier based on the expression levels of two genes, G1 and G2:

Training data set:
- Orange: Obama
- Blue: McCain
- Black: Unknown

How do we do this algorithmically?
- Linear vs. nonlinear
- Multiple dimensions...?
Recipe for classification

1. Choose a classification method

2. Feature selection / dimension reduction

3. Train the classifier

4. Assess classifier performance
1. Choose a classification method

linear:
  – Linear Discriminant Analysis (LDA)
  – Nearest Centroid

nonlinear:
  – k Nearest Neighbors (kNN)
  – Artificial Neural Network (ANN)
  – Support Vector Machine (SVM)
Linear Discriminant Analysis (LDA)

Find a line / plane / hyperplane

assumptions:
• sampled from a normal distribution
• variance/covariance same in each class

R: `lda` (*MASS* package)
Nearest centroid

Calculate centroids for each class.

Similar to LDA.

Can be extended to multiple (n > 2) classes.
k Nearest Neighbors (kNN)

For a test case, find the $k$ nearest samples in the training set, and let them vote.

- need to choose $k$ (hyperparameter)
- $k$ must be odd
- need to choose distance metric

No real “learning” involved - the training set defines the classifier.

R: `knn` (class package)
Artificial Neural Network (ANN)

Each “neuron” is simply a function (usually nonlinear).

The “network” to the left just represents a series of nested functions.

Interpretation of results is difficult.

R: nnet (nnet package)
Support Vector Machine (SVM)

Find a line / plane / hyperplane maximizing margin. Uses the “kernel trick” to map data to higher dimensions.

• very flexible
• many parameters

Results can be difficult to interpret.

R: `svm` (e1071 package)
## Comparison of methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Linear discriminant analysis</th>
<th>Nearest centroid</th>
<th>KNN</th>
<th>Neural networks</th>
<th>Support vector machines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple method</td>
<td>Based on distance calculation</td>
<td>Good for simple problems</td>
<td>Advanced methods</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Good for few training samples</td>
<td>Distribution of data assumed</td>
<td></td>
<td>Involve machine learning</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Several adjustable parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Many training samples required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(e.g., 50-100)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Flexible methods</td>
<td></td>
</tr>
</tbody>
</table>

![Graph showing data distributions in G1 and G2]
Recipe for classification

1. Choose a classification method

2. Feature selection / dimension reduction

3. Train the classifier

4. Assess classifier performance
Dimension reduction - why?

Previous slides: 2 genes X 16 samples.

Your data: **22000** genes X 31 samples.

More genes = more parameters in classifier. **Potential for over-training!!** aka *The Curse of Dimensionality.*
Dimension reduction

Two approaches:

1. Data transformation
   - Principal component analysis (PCA); use top components
   - Average co-expressed genes (e.g. from cluster analysis)

2. Feature selection (gene selection)
   - Significant genes: t-test / ANOVA
   - High variance genes
   - Hypothesis-driven
Recipe for classification

1. Choose a classification method

2. Feature selection / dimension reduction

3. Train the classifier

4. Assess classifier performance
Train the classifier

If you know the exact classifier you want to use: **just do it.**

but...

Usually, we want to try several classifiers or several variations of a classifier. e.g. number of features, $k$ in kNN, etc.

The problem:

Testing several classifiers, and then choosing the **best one** leads to selection bias (= **overtraining**). This is bad.

Instead we can use **cross-validation** to choose a classifier.
Cross validation

Data set

Temporary training set

Train Classifier

Temporary testing set

Apply classifier to temporary testing set

Performance

Do this several times!

Each time, select different sample(s) for the temporary testing set.

Leave-one-out cross-validation (LOOCV):
Do this once for each sample (i.e. the temporary testing set is only the one sample).

Assess average performance.

If you are using cross-validation as an optimization step, choose the classifier (or classifier parameters) that results in best performance.
Cross-validation example

Given data with 100 samples

5-fold cross-validation:
  Training: 4/5 of data (80 samples)
  Testing: 1/5 of data (20 samples)
  - 5 different models and performance results

Leave-one-out cross-validation (LOOCV)
  Training: 99/100 of data (99 samples)
  Testing: 1/100 of data (1 sample)
  - 100 different models
Recipe for classification

1. Choose a classification method

2. Feature selection / dimension reduction

3. Train the classifier

4. Assess classifier performance
Assess performance of the classifier

How well does the classifier perform on *novel* samples?

Worst: Assess performance on the training set.


Alternative: Set aside a subset of your data as a test set.

Best: *Also* assess performance on an entirely independent data set (e.g. data produced by another lab).
Problems with cross-validation

1. You cannot use the same cross-validation to optimize and evaluate your classifier.

2. The classifier obtained at each round of cross-validation is different, and is different from that obtained using all data.

3. Cross-validation will overestimate performance in the presence of experimental bias.
Assessing classifier performance

Predict relapse of cancer

<table>
<thead>
<tr>
<th></th>
<th>prediction</th>
<th>actual outcome</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aron</td>
<td>yes</td>
<td>no</td>
<td>FP</td>
</tr>
<tr>
<td>Beth</td>
<td>no</td>
<td>no</td>
<td>TN</td>
</tr>
<tr>
<td>Chuck</td>
<td>yes</td>
<td>yes</td>
<td>TP</td>
</tr>
<tr>
<td>Dorthe</td>
<td>no</td>
<td>yes</td>
<td>FN</td>
</tr>
</tbody>
</table>
| ...     | ...        | ...            | ...

TP = True Positive  
TN = True Negative  
FP = False Positive  
FN = False Negative

Confusion matrix

<table>
<thead>
<tr>
<th>predict yes</th>
<th>predict no</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual yes</td>
<td>131 TP</td>
</tr>
<tr>
<td>actual no</td>
<td>7 FP</td>
</tr>
</tbody>
</table>
Classifier performance: Accuracy

\[ \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \]

fraction of predictions that are correct

range: \([0 \ .. \ 1]\)

confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>predict yes</th>
<th>predict no</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual yes</td>
<td>131 TP</td>
<td>21 FN</td>
</tr>
<tr>
<td>actual no</td>
<td>7 FP</td>
<td>212 TN</td>
</tr>
</tbody>
</table>

Accuracy = 0.92
Unbalanced test data

a new classifier: does patient have tuberculosis?

confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>predict yes</th>
<th>predict no</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual yes</td>
<td>5345 TP</td>
<td>21 FN</td>
</tr>
<tr>
<td>actual no</td>
<td>113 FP</td>
<td>18 TN</td>
</tr>
</tbody>
</table>

Accuracy = 0.98 (?)

For patients who do not really have tuberculosis: this classifier is usually wrong!

>>> Accuracy can be misleading, especially when the test cases are unbalanced
Classifier performance: sensitivity/specificity

\[
\text{Sensitivity} = \frac{TP}{TP + FN}
\]

\[
\text{Specificity} = \frac{TN}{TN + FP}
\]

- Sensitivity: the ability to detect positives
- Specificity: the ability to reject negatives

Confusion matrix:

<table>
<thead>
<tr>
<th></th>
<th>predict yes</th>
<th>predict no</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual yes</td>
<td>131 TP</td>
<td>21 FN</td>
</tr>
<tr>
<td>actual no</td>
<td>7 FP</td>
<td>212 TN</td>
</tr>
</tbody>
</table>

Sensitivity = 0.86
Specificity = 0.97
Classifier performance: Matthews Correlation Coefficient

$$MCC = \frac{(TP \cdot TN) - (FN \cdot FP)}{\sqrt{(TN + FN)(TN + FP)(TP + FN)(TP + FP)}}$$

A single performance measure that is less influenced by unbalanced test sets

range: [-1 .. 1]

confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>predict yes</th>
<th>predict no</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual yes</td>
<td>131 TP</td>
<td>21 FN</td>
</tr>
<tr>
<td>actual no</td>
<td>7 FP</td>
<td>212 TN</td>
</tr>
</tbody>
</table>

MCC = 0.84
Caution: unbalanced test data

a new classifier: does patient have tuberculosis?

confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>predict yes</th>
<th>predict no</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual yes</td>
<td>5345 TP</td>
<td>21 FN</td>
</tr>
<tr>
<td>actual no</td>
<td>113 FP</td>
<td>18 TN</td>
</tr>
</tbody>
</table>

Accuracy = 0.98 (?)

Sensitivity = 0.996
Specificity = 0.14
MCC = 0.24
Unbiased classifier assessment

Avoid information leakage!

1. Use separate data sets for training and for testing.

2. Try to avoid “wearing out” the testing data set. Testing multiple classifiers (and choosing the best one) can only increase the estimated performance -- therefore it is biased!

3. Perform any feature selection using only the training set (or temporary training set).
Summary: Recipe for classification

1. Choose a classification method
   – kNN, LDA, SVM, etc.

2. Feature selection / dimension reduction
   – PCA, possibly t-tests

3. Train the classifier
   – use cross-validation to optimize parameters

4. Assess classifier performance
   – use an independent test set if possible
   – determine sensitivity and specificity when appropriate
The data set in the exercise

Genome-wide expression profiling of human blood reveals biomarkers for Huntington's disease.


31 samples:
- 14 normal
- 5 presymptomatic
- 12 symptomatic

Affymetrix HG-U133A arrays (and Codelink Uniset)
The data set in the exercise

Huntingon’s disease
- neurological disorder
- genetic
- polyglutamine expansion in huntingtin gene

Why search for marker of disease progression (not diagnosis)
- assess treatment efficacy
- surrogate endpoint in drug trials
Questions and/or Lunch