Machine learning with Bioconductor

M. T. Morgan (mtmorgan@fhcrc.org)

Fred Hutchinson Cancer Research Center
Seattle, WA

http://bioconductor.org
10 October 2006

Overview

A machine learning checklist

- Filter (see lab)
- Feature selection
- Metrics: distance measures
- Learn: un-supervised & supervised
- Assess: cross-validation & beyond

Distance measures

Packages: dist, bioDist, daisy, ...

Typical distance measures (e.g., bioDist)

- `euc`: squared distance between two vectors; sensitive to scale
- `cor.dist`: correlation (i.e., variance-standardized), so approximately scale-invariant
- `spearman.dist`, `tau.dist`: rank-based correlation, so more robust
- `mutualInfo, MIdist`: binned, then mutual information

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}$$

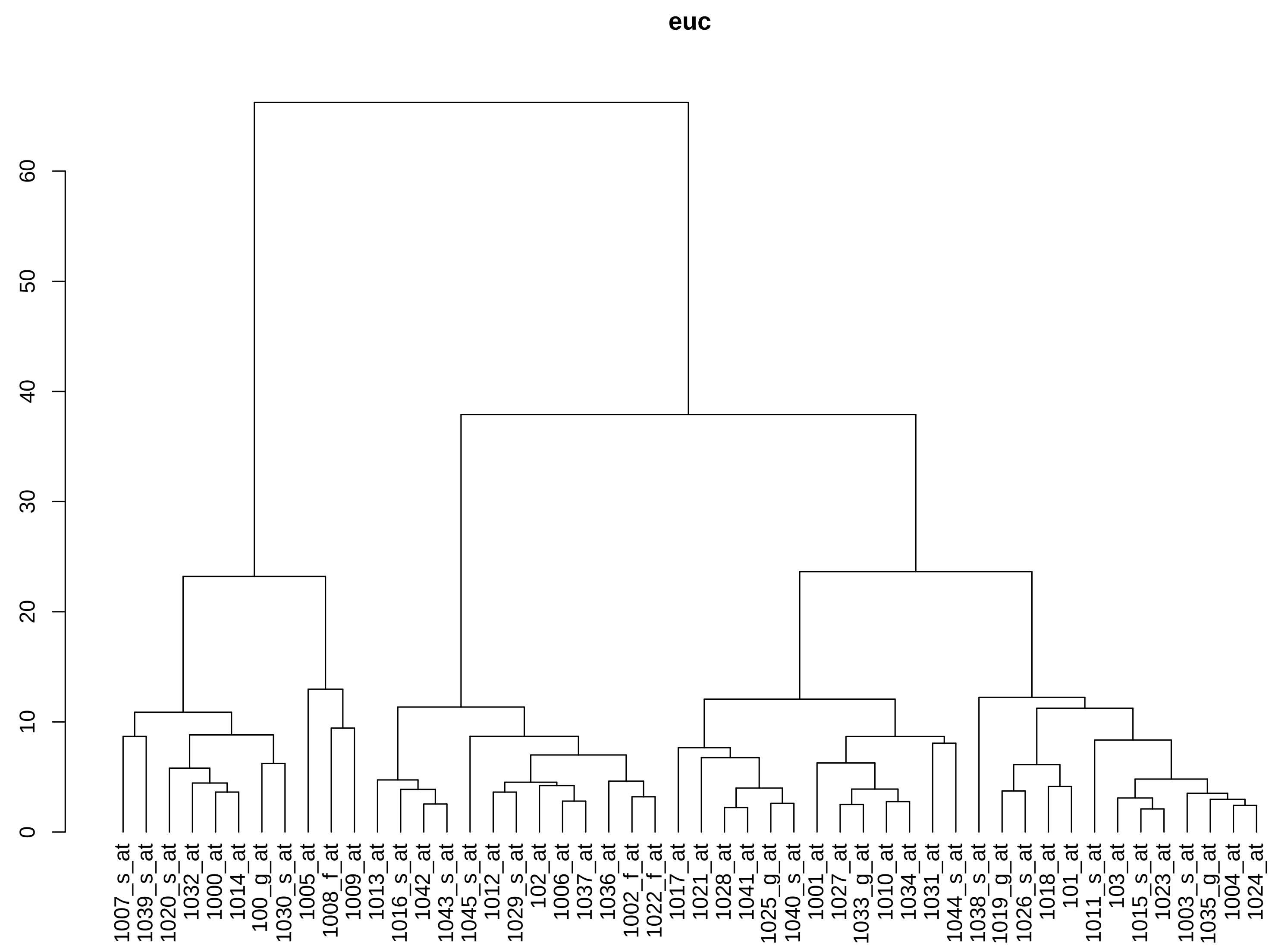
- `man`: ‘Manhattan’ distance

Euclidean distances

```
> library("Biobase")
> library("bioDist")
> library("ALL")
> data(ALL)
> allSubset = ALL[1:50, ALL$mol.biol %in%
+     c("BCR/ABL", "NEG")]
> allSubset$mol.biol <- factor(allSubset$mol.biol)
> eucDistance <- euc(allSubset)
```

Summarize, plot, and interpret…

```
> eucClust <- hclust(eucDistance)
> plot(as.dendrogram(eucClust), main = "euc")
```


euc
0
10
20
30
40
50
60
1007_s_at
1039_s_at
1020_s_at
1032_at
1000_at
1014_at
100_g_at
1030_s_at
1005_at
1008_f_at
1009_at
1013_at
1016_s_at
1042_at
1043_s_at
1045_s_at
1012_at
1029_s_at
102_at
1006_at
1037_at
1036_at
1002_f_at
1022_f_at
1017_at
1021_at
1028_at
1041_at
1025_g_at
1040_s_at
1001_at
1027_at
1033_g_at
1010_at
1034_at
1031_at
1044_s_at
1038_s_at
1019_g_at
1026_s_at
1018_at
101_at
1011_s_at
103_at
1015_s_at
1023_at
1003_s_at
1035_g_at
1004_at
1024_at

Distance metrics matter

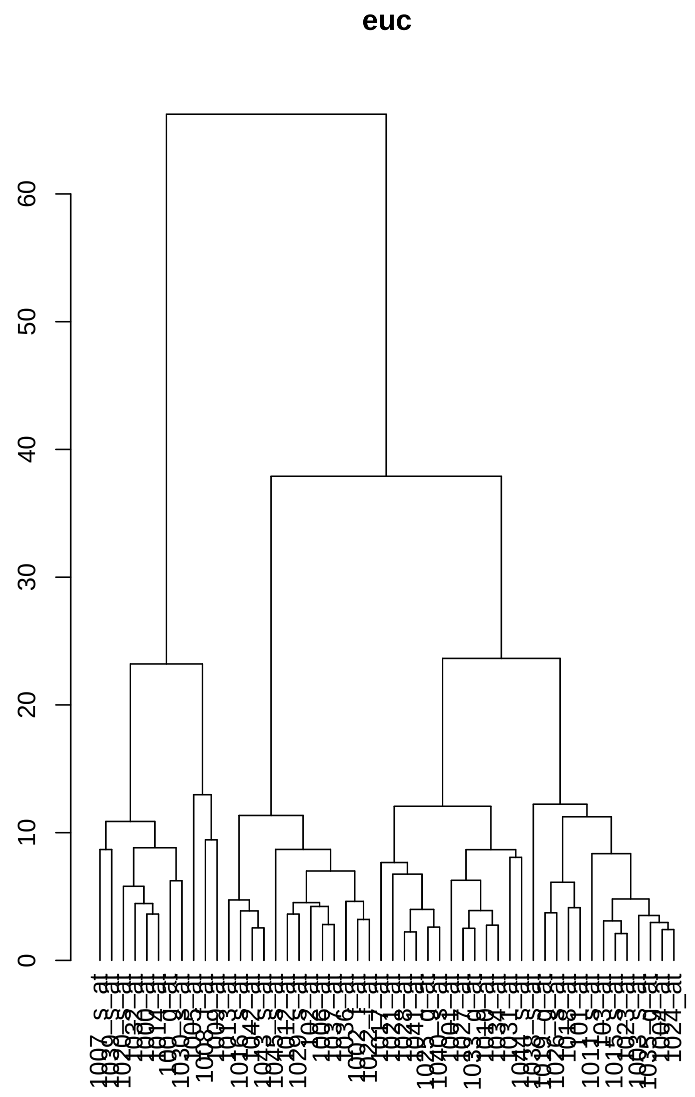
- `euc` measures Euclidean distance; sensitive to measurement scale
- Between-gene expression values can be quite heterogenous

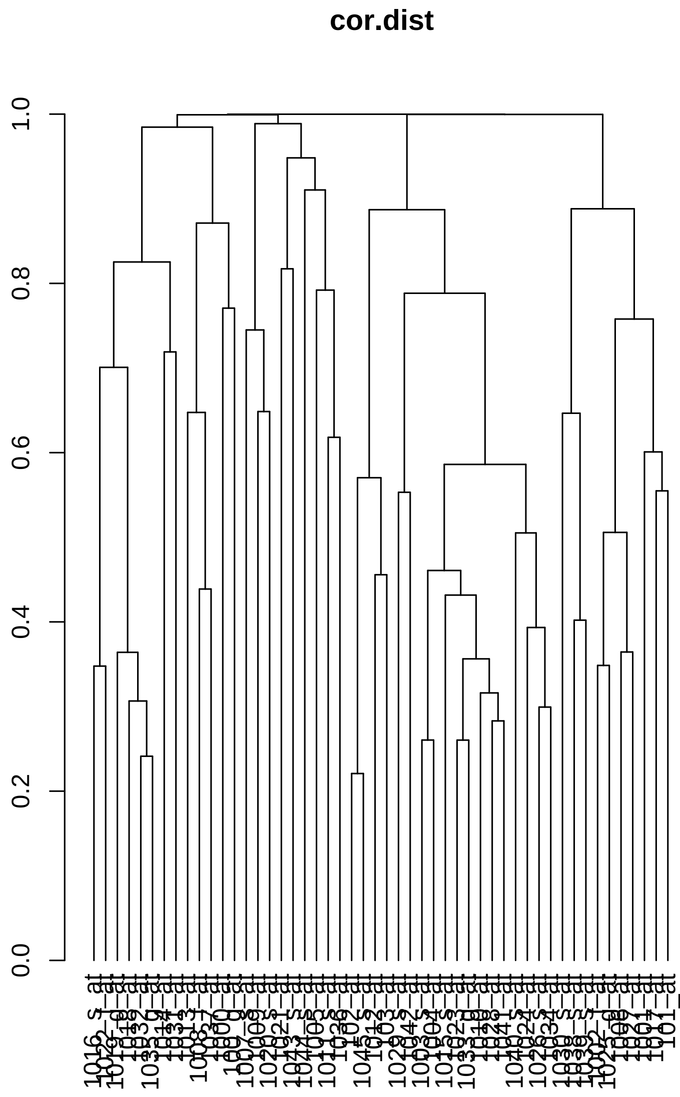
```
> summary(apply(exprs(allSubset), 1, mean))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.042   4.049   5.456   5.523   6.424   9.311

> summary(apply(exprs(allSubset), 1, var))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.02291 0.05178 0.07497 0.16850 0.21710 1.22200
```

Scale-independent distances

Different from euclidean distances?

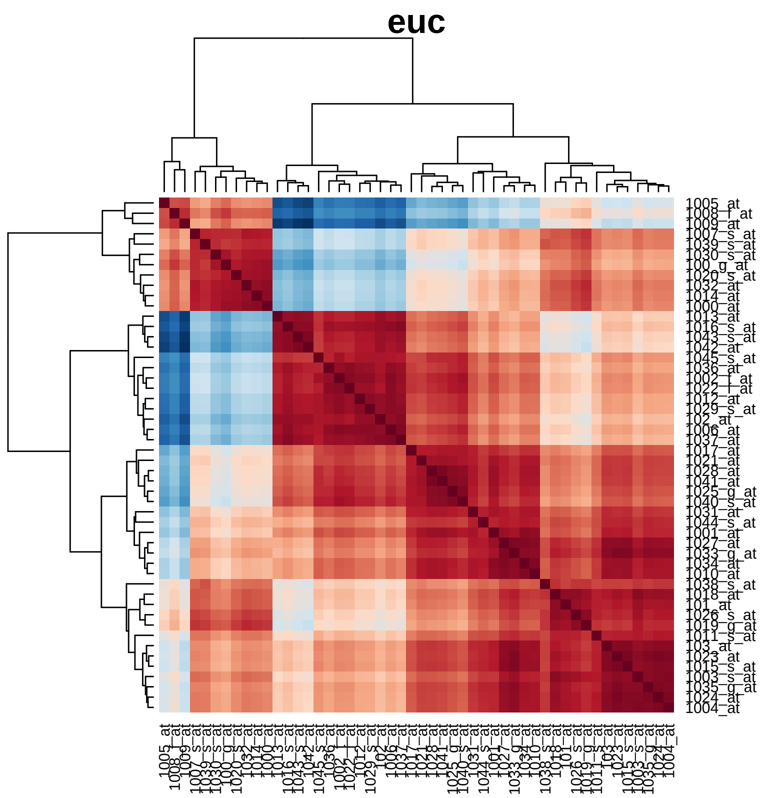
```
> originalOptions <- par(mfrow = c(1, 2))
> eucClust <- hclust(euc(allSubset))
> plot(as.dendrogram(eucClust), main = "euc")
> corClust <- hclust(cor.dist(allSubset))
> plot(as.dendrogram(corClust), main = "cor.dist")
> par(originalOptions)
```


euc
0
10
20
30
40
50
60

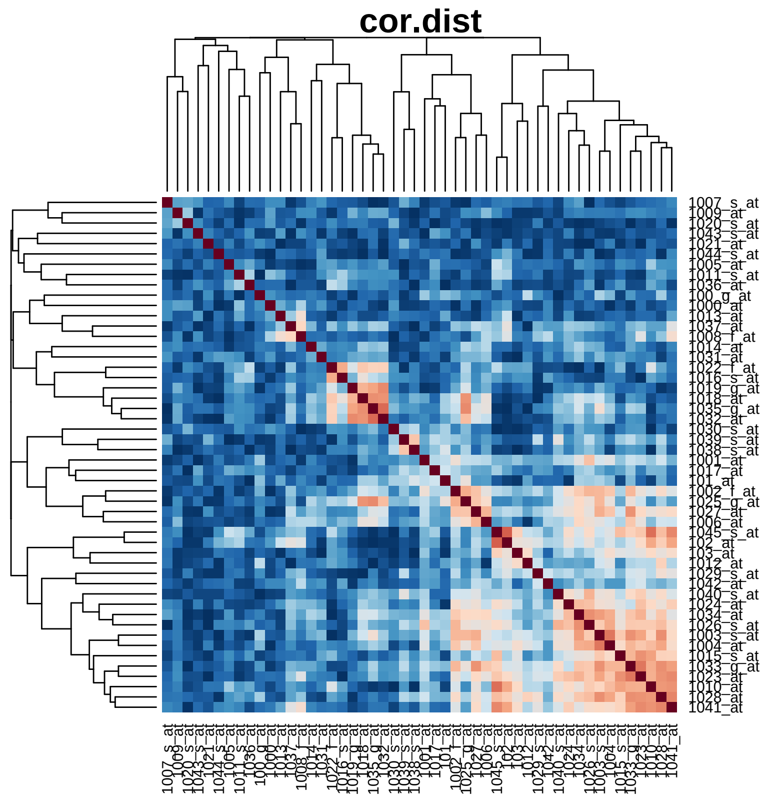

cor.dist
0.0
0.2
0.4
0.6
0.8
1.0

Visualizing dendrogram structure

```
> eucMatrix <- as.matrix(euc(allSubset))
> heatmap(eucMatrix, symm = TRUE, col = heatmapColor,
+     distfun = as.dist, main = "euc")
```



```
> corMatrix <- as.matrix(cor.dist(allSubset))
> heatmap(corMatrix, symm = TRUE, col = heatmapColor,
+     distfun = as.dist, main = "cor.dist")
```



Options for subsequent analysis

- Choose appropriate distance metric, if algorithm permits
- Transform data prior to measuring distance

```
> exprs(allSubset) <- t(apply(exprs(allSubset),
+     1, scale))
```

Better options indicated in the lab!

Machine learning

- Methods of inference to create algorithms for prediction (classification of new samples)

Major types of machine learning

- *Unsupervised*: no prior information on classification outcome, e.g., clustering. Implicit in visualization of distance metrics
- *Supervised*: *a priori* information (such as tumor status) on classification

Supervised machine learning

Overall scenario

- Use existing data with information on gene expression levels and phenotypes to devise an algorithm to classify samples with unknown phenotype

```
> levels(allSubset$mol.biol)
[1] "BCR/ABL" "NEG"
```

Steps

- Apply non-specific filters to identify informative genes
- Develop the classification algorithm
- Assess performance of classification algorithm, typically using *cross-validation*

Machine learning algorithms

Linear algorithms

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

- x: sample; w: weights determined during training, w_0: threshold for classification
- 'Linear' indicates linear combination of features
- Adjust weights to 'best' assign samples to their *a priori* types
- Weights represent estimable parameters, and sample size limits the number of estimable parameters
- E.g., linear discriminant analysis

Machine learning algorithms (continued)

- *Non-linear*, e.g., neural networks
- *Regularized*, e.g., support vector machines
- *Local*, e.g., k nearest neighbor
- *Tree-based*, e.g., classification and regression tree (CART)

MLInterfaces

```
> library(MLInterfaces)
```

- Unified interface to many machine learning algorithms
- Interface provided for…

class	`knn1, knn.cv, lvq1, lvq2, lvq3, olvq1, som`
	`SOM`
cluster	`agnes, clara, diana, fanny, silhouette`
e1071	`bclust, cmeans, cshell, hclust, lca`
	`naiveBayes, svm`
gbm	`gbm`
ipred	`bagging, ipredknn, lda, slda`
MASS	`isoMDS, qda`
nnet	`nnet`
pamr	`cv, knn, pam, pamr`
randomForest	`randomForest`
rpart	`rpart`
stats	`kmeans`

Developing a machine learning algorithm

- Divide sample into *training* and *test* sets
- Identify an *a priori* classification
- Use training set to develop a specific algorithm
- Use test set to assess algorithm performance

```
> result <- knnB(allSubset, classifLab = "mol.biol",
+     trainInd = 1:41)
```

`knnB`

- Invokes function `knn`, provided by package class
- Distance metric: Euclidean

Summarize test classifications with a *confusion matrix*:

```
> confuMat(result)

         predicted
given      BCR/ABL NEG
  BCR/ABL        7   8
  NEG           30  25
```

Model assessment with cross-validation

A great diversity of machine learning algorithms

- Which is 'best'?

What is 'best'?

- Ability to correctly classify new samples?
- Minimize uncertainty of each classification?

No free lunch: all models are best, in the domain of their assumptions

Assessing model performance

A quandary:

- New samples are not already classified, so how can we know when our algorithm is working?

Solution:

- Divide sample into *training* and *test* sets
- Identify an *a priori* classification
- Use training set to develop a specific algorithm
- Use test set to assess algorithm performance

```
> result <- knnB(allSubset, classifLab = "mol.biol",
+     trainInd = 1:41)
```

Cross-validation

- *Repeatedly* divide data into training set and test set, and assess algorithm performance
- Several ways to divide data: leave-one-out, leave-out-group, etc.

Leave-one-out cross-validation

- All but 1 sample included in the training set
- Assess performance of trained algorithm based on classification (correct or not) of remaining sample
- Repeat for all possible training sets: if there are $n = 100$ samples, then there are $n = 100$ cross-validations

Cross-validation with `xval`

```
> allKnnXval <- xval(allSubset, classLab = "mol.biol",
+     proc = knnB, xvalMethod = "LOO")
> length(allKnnXval)

[1] 111

> allKnnXval[1:4]

[1] "NEG" "NEG" "NEG" "NEG"
```

- `xvalMethod`: leave-one-out (`LOO`), but others possible
- Result is a character vector; each element represents one cross-classification, indicating how the *i*th individual was classified when left out

Assessing model fit

How well was each sample classified?

```
> as.character(allSubset$mol.biol[1:4])

[1] "BCR/ABL" "NEG"     "BCR/ABL" "NEG"

> allKnnXval[1:4]

[1] "NEG" "NEG" "NEG" "NEG"

> table(given = allSubset$mol.biol, predicted = allKnnXval)

         predicted
given      BCR/ABL NEG
  BCR/ABL       17  20
  NEG           26  48
```

Feature selection

- Problem: sample size sets an upper limit on the number of features that can be used in a classification algorithm
- Solution: reduce number of features, without using knowledge of classification ability, to those that are most informative
- Must be applied consistently to each cross-validation

```
> library(genefilter)

Loading required package: survival
Loading required package: splines
```

Implementing feature selection

```
> allSubset = ALL[, ALL$mol.biol %in% c("BCR/ABL",
+     "NEG")]
> allSubset$mol.biol <- factor(allSubset$mol.biol)
> exprs(allSubset) <- t(apply(exprs(allSubset),
+     1, scale))
> tSelection <- function(data, classifier) {
+     tTests <- rowttests(data, data[[classifier]],
+         tstatOnly = FALSE)
+     abs(tTests$statistic)
+ }
> tStats <- tSelection(allSubset, "mol.biol")
> tTop50 <- order(tStats, decreasing = TRUE)[1:50]
```

Implementing feature selection (continued)

Any improvement with a single set of training individuals?

```
> confuMat(knnB(allSubset[tTop50, ], classifLab = "mol.biol",
+     trainInd = 1:41))

         predicted
given      BCR/ABL NEG
  BCR/ABL       14   1
  NEG            0  55

> confuMat(knnB(allSubset[1:50, ], classifLab = "mol.biol",
+     trainInd = 1:41))

         predicted
given      BCR/ABL NEG
  BCR/ABL        7   8
  NEG           30  25
```

Feature selection in *each* cross-validation

```
> tTopKnnXval <- xval(allSubset, "mol.biol",
+     knnB, "LOO", group = 0:0, fsFun = tSelection,
+     fsNum = 50)
> table(given = allSubset$mol.biol, predicted = tTopKnnXval[["out
```

```
         predicted
given      BCR/ABL NEG
  BCR/ABL       31   6
  NEG            1  73
```

```
> table(given = allSubset$mol.biol, predicted = allKnnXval)
```

```
         predicted
given      BCR/ABL NEG
  BCR/ABL       17  20
  NEG           26  48
```

Recap

- Distance metrics are very important
- Diverse machine learning algorithms available
- Cross-validation assesses algorithm performance
- Feature selection reduces number of features to a (statistically and computationally) reasonable number

Directions

Machine learning

- Assessing feature importance, e.g., assessing consequences of feature permutation in test sets with several samples
- edd: use machine learning to choose between different models (e.g., unimodal; bimodal) describing the relationship between features and phenotypes
- …

More generally…

- Extensive opportunity for rigorous, creative analysis in Bioconductor (e.g., limma, for linear models) and R