

Machine learning for high-throughput biology: Approaches in Bioconductor

VJ Carey, Ph.D.
Longwood March 2008

- Some recent ambitions
- Sample classification problem; basic terms
- Bioconductor MLInterfaces design
- Illustrations

Some current ambitions: microRNA target prediction

M.Yousef et al.

```
3' uagcgccaaauauggUUUACUUA 5' has-miR-579
   ||: | ||||: ||||:|
5' atttctttttatggaAAATGAGT 3' LR1G3
   out-seed          seed
```

Fig. 1. Duplex partitioned into two parts, the seed and the out-seed, for miRNA hsa-miR-579 and its target LRIG3. The seed part is shown in capital letters.

Bioinformatics Nov 2007

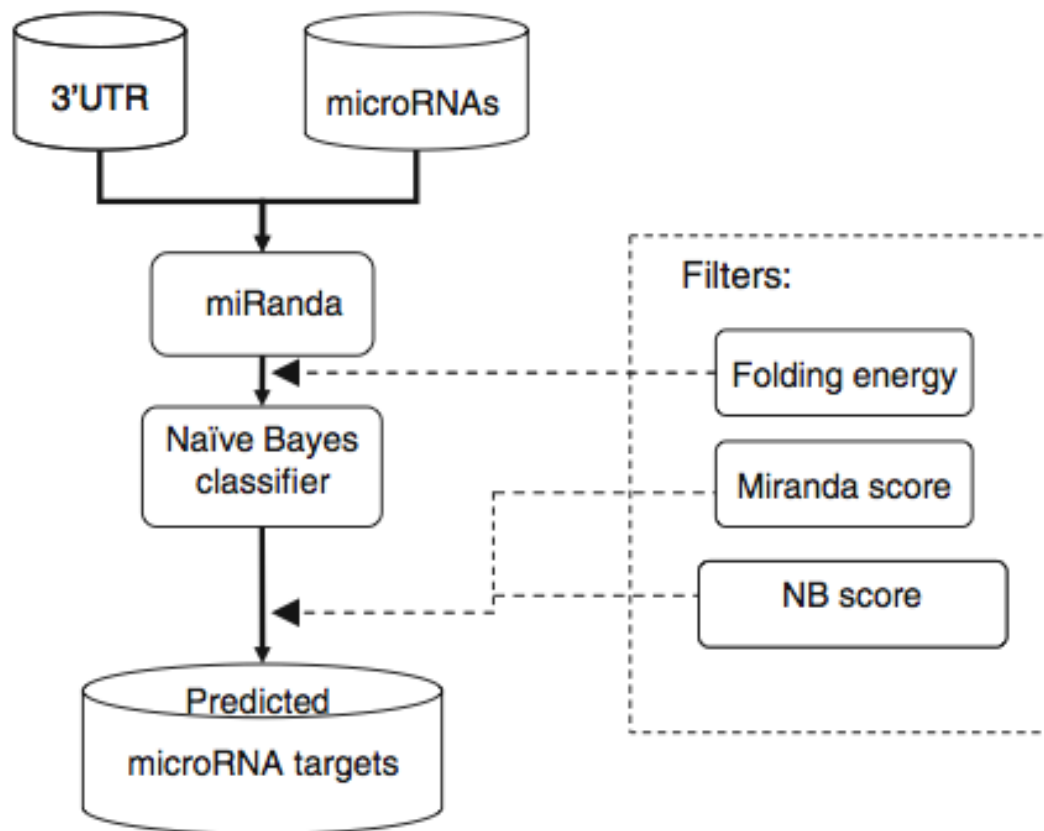


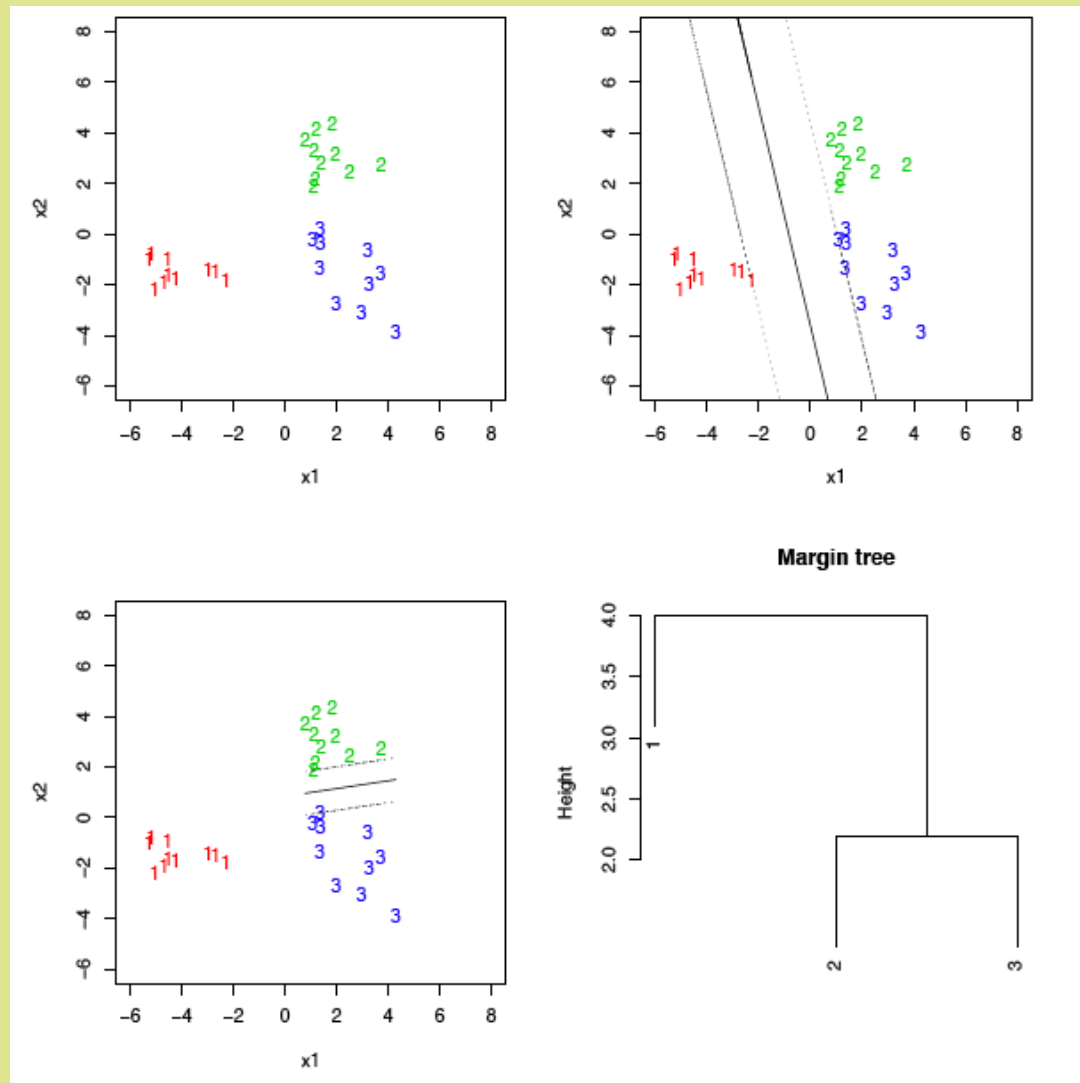
Fig. 2. The computational procedure for implementing NBmiRTar tool for miRNA target prediction.

Features for target prediction – mostly counts!

Table 1. The top 30 significant features obtained by MI

| | Feature name | Positive Mean | Negative Mean |
|----|---|---------------|---------------|
| 1 | Number of bulges in seed | 0.35 | 0.004 |
| 2 | Number of bulges in seed with length 1 | 0.328 | 0.0035 |
| 3 | Number of symmetric loops in out-seed with length 1 | 1.186 | 2.718 |
| 4 | Number of bulges in out-seed | 0.764 | 1.655 |
| 5 | Number of asymmetric loops in out-seed with length greater than 7 | 0.15 | 0.0095 |
| 6 | Acucca | 0.0755 | 0 |
| 7 | Acuc | 0.128 | 0.0085 |
| 8 | Number of asymmetric loops in seed | 0.0711 | 0 |
| 9 | Acucc | 0.111 | 0.006 |
| 10 | Number of asymmetric loops in out-seed | 1.04 | 0.4835 |
| 11 | Accuu | 0.084 | 0.002 |
| 12 | Gcuuu | 0.057 | 0 |
| 13 | Gcuuua | 0.057 | 0 |

Some current ambitions: structured multiclass prediction



Tibshirani and Hastie 2007 JMLR

Some current ambitions: open source discipline

1. Reproducibility of scientific research is increased
2. Algorithms implemented in same framework facilitate fair comparisons
3. Problems can be uncovered much faster
4. Bug fixes and extensions from external sources
5. Methods are more quickly adopted by others
6. Efficient algorithms become available
7. Leverage existing resources to aid new research
8. Wider use leads to wider recognition
9. More complex machine learning algorithms can be developed
10. Accelerates research
11. Benefits newcomers and smaller research groups

Table 3: Eleven Advantages of Machine Learning Open Source Software

Sonnenburg et al, 2007 JMLR

Summary/Road map

- accurate predictive modeling can be useful in many bioinformatic domains
- currently, “large margin” classifiers attract attention; also “boosting”
- optimality of classifier selection not readily verified; may be illusory (see, e.g., No Free Lunch Theorem in Duda, Hart Stork)
- feature selection and representation choices may require human insight
- in this talk, we focus on HOW TO DO various types of machine learning
- you will have to answer the WHY questions in your own applications

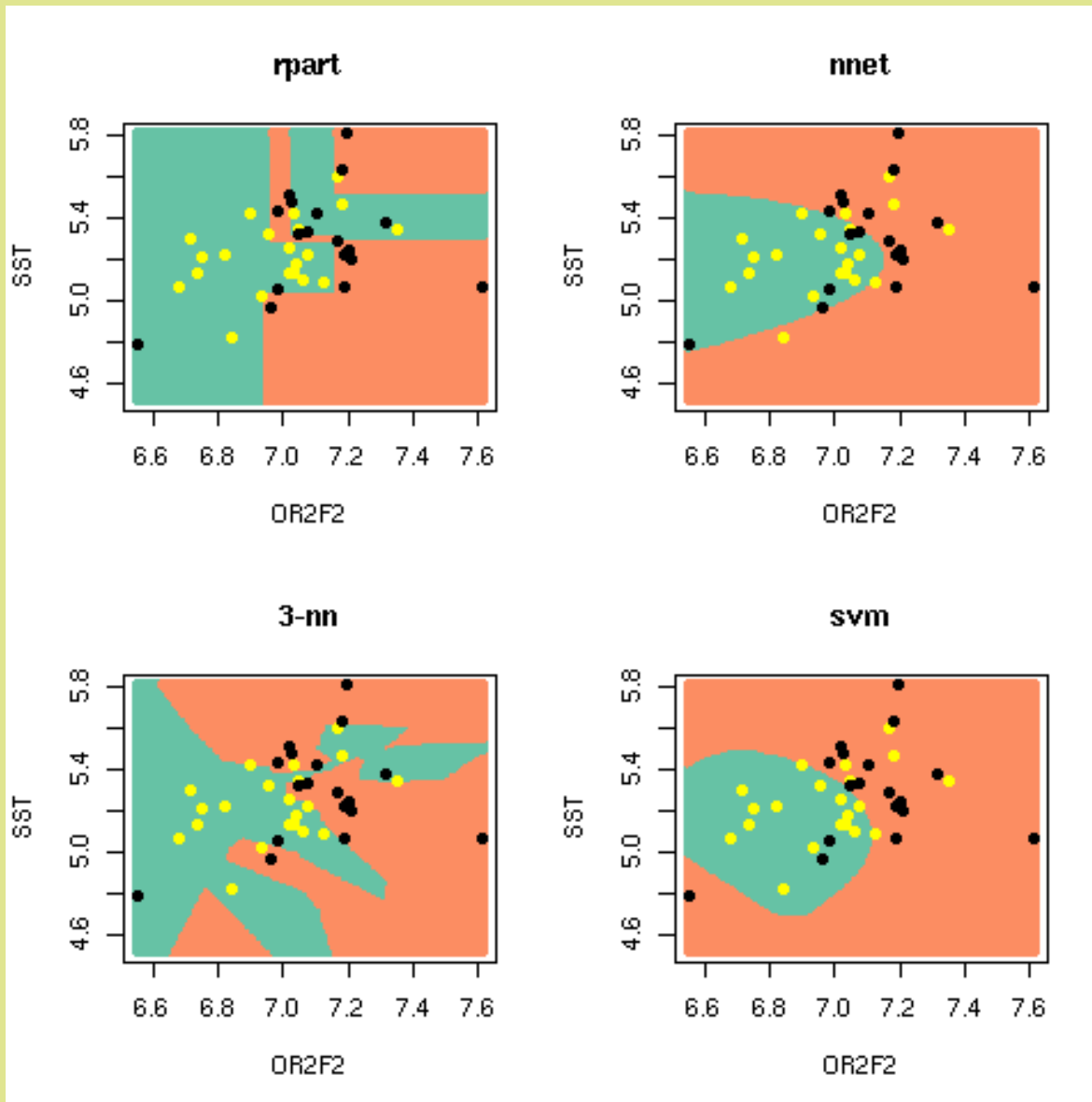
Sample classification problem

- 79 cases of B-Cell Acute Lymphocytic Leukemia; U95Av2 chips, filtered by MAD
- Aim: understand transcriptional signature of BCR/ABL fusion

```
> nsALLb
ExpressionSet (storageMode: lockedEnvironment)
assayData: 1263 features, 79 samples
  element names: exprs
phenoData
  rowNames: 01005, 01010, ..., 84004 (79 total)
  varLabels and varMetadata description:
    cod: Patient ID
    diagnosis: Date of diagnosis
    ....: ...
    date last seen: date patient was last seen
    (21 total)
featureData
  featureNames: 1005_at, 1038_s_at, ..., AFFX-M27830_5_at (1263 total)
  fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
pubMedIds: 14684422 16243790
Annotation: hgu95av2

> table(nsALLb$mol.biol)
BCR/ABL      NEG
    37      42
```

Result for two-gene predictor: Painting in feature space



Cautions

- Prediction typically involves extrapolation to points where no data were seen
- Sharp boundaries may mask genuine ambiguity – when posterior probabilities of class membership are available, use them to measure confidence of assertion
- Selection of tuning parameter values can profoundly influence performance
 - pruning of tree
 - size/regularization of nnet
 - choice of (k, l) in k -nn
 - choice of kernel and costs in SVM
- Feature selection, if used, must be a part of cross-validatory assessment

Basic theory of the problem

- Formalism:
 - **population** of N objects is partitioned by class labels in a set C ; proportion of objects in class $k \in C$ is π_c
 - each **object** $x_i, i \in 1, \dots, N$ is identified with a p -dimensional **raw feature vector** and the **class label** $c_i \in C$
- **objective of classification:** use the data to define a rule $c(\cdot)$ that satisfies $c_i = c(x_i)$
- **implementation:** select and try to minimize a **loss function** measuring the discrepancy between $c(x)$ and the true class of the object x , penalize for roughness of solution
- **figures of merit:** Bayes error rate, generalization error sensitivity, specificity, relative efficiency
- estimators: test set error, CV-error, OOB error rate

major concerns for bioinformatics

- $p \gg N$, so it is relatively easy to find perfect classifiers in finite data, especially if raw features can be transformed
- problems are suspected to be **sparse**: most features are irrelevant (coefficient zero in linear models) but we don't know which! "automatic feature selection" of interest
- are we satisfied with a predictive a black box, or do we want insight into relationships between features and class values ?
- interpretable parameters are useful when reasonable properties can be asserted for their estimators

some broad options

Simplest data setup

- the labels are $c \in \{0, 1\}$, with population class proportions π_0 and π_1
- features are $x_i \in R^p$
- in other cases, e.g., given features are sequence, choice of feature representation and similarity measure are fundamental determinants of process

-
- Option A: model $Pr(c_i = 1|x_i)$ and use this model to define the prediction function g – examples are logistic discrimination, neural nets

- Option B: model $x_i|c_i \sim N(\mu_{c_i}, \Sigma_{c_i})$

LDA: assuming common covariance Σ , allocate x_i to class 1 when

$$LD(x_i) = (\mu_1 - \mu_0)^t \Sigma^{-1} (x_i - .5(\mu_1 + \mu_2)) > \log(\pi_0/\pi_1)$$

- Option C: non-parametric approaches: algorithmic linear discriminant; CART/random forest/boosting, k -NN

a view of two cultures (suggested by Breiman, 2001 Stat Sci with discussion)

- statistical modelers: analysis of **mass behavior** in a dataset leads to inference on distinctness of groups, desire for explanatory power
- algorithmic modelers: (computational learning theory) algorithms for identifying **boundary cases** lead to criteria for distinguishing data elements into groups; explanation secondary to decision/prediction; black boxes tolerated

this idea of downweighting data far from the class boundary seems central to boosting and svm

Fundamental terms/performance measures

Bayes classifier: Let $p(k|x)$ be the posterior probability that an object with features x is of class k ; let $L(k, l)$ denote cost of classifying instance of class k as an l , and let d denote cost of declaring doubt.

$$c(x) = \begin{cases} k & \text{if } k \text{ attains } \min_{l \in \{1, \dots, K\}} \sum_j L(j, l)p(j|x) < d \\ \mathbf{D} & \text{otherwise} \end{cases}$$

Under 0-1 loss, no doubt option, the Bayes classifier classifies to $\max_{l \in \{1, \dots, K\}} p(l|x)$.

Bayes error rate: probability of misclassification in the presence of full information on distribution of features given classes and prior probabilities of classes

True error rate of a rule $c(x)$: rate at which c misclassifies objects in the population

Apparent error rate of a rule $c(x)$: rate at which c was observed to misclassify elements of a training set; observed misclassification rate

Confusion matrix: cross tabulation of true class against predicted class

Cross-validated error rate estimate: average of observed misclas-

sification rates over a series of training/test data splits

Out-of-bag error rate: average of observed misclassification rates over a series of bootstrap iterations; observations excluded in the bootstrap sample are 'out of bag'

Setup

```
> library(ALL)
> library(MLInterfaces)
> if (!exists("ALL")) data(ALL)
> library(genefilter)
> mads = apply(exprs(ALL), 1, mad)
> ok = which(mads > quantile(mads, 0.9))
> nsALL = ALL[ok, ]
> nsALLb = nsALL[, nsALL$mol.biol %in% c("NEG", "BCR/ABL")]
> nsALLb$mol.biol = factor(nsALLb$mol.biol)
> isb = which(substr(nsALLb$BT, 1, 1) == "B")
> nsALLb = nsALLb[, isb]
```

Using these concepts with the ALL data

```
> library(MLInterfaces)
> ld1 = MLearn(mol.biol~., nsALLb, ldaI,
+             xvalSpec("LOG", 5, balKfold.xvspec(5)))
> ld1
```

MLInterfaces classification output container

The call was:

```
MLearn(formula = mol.biol ~ ., data = nsALLb, method = ldaI,
       trainInd = xvalSpec("LOG", 5, balKfold.xvspec(5)))
```

Predicted outcome distribution for test set:

| BCR/ABL | NEG |
|---------|-----|
| 38 | 41 |

history of feature selection in cross-validation available; use fsHi

```
> confuMat(ld1)
```

| | predicted | |
|-------|-----------|-----|
| given | BCR/ABL | NEG |

| | | |
|---------|----|----|
| BCR/ABL | 32 | 5 |
| NEG | 6 | 36 |

The out-of-bag error rate with randomForest

```
> rf1 = MLearn(mol.biol~., nsALLb, randomForestI,  
+             xvalSpec("NOTEST"))  
  
> RObject(rf1)
```

Call:

```
randomForest(formula = formula, data = trdata)  
              Type of random forest: classification  
              Number of trees: 500  
No. of variables tried at each split: 35
```

OOB estimate of error rate: 15.19%

Confusion matrix:

| | BCR/ABL | NEG | class.error |
|---------|---------|-----|-------------|
| BCR/ABL | 30 | 7 | 0.1891892 |
| NEG | 5 | 37 | 0.1190476 |

Using naive Bayes with cross-validation

```
> x5 = xvalSpec("LOG", 5, balKfold.xvspec(5))
> zz = MLearn(mol.biol~., data=nsALLb,
+           naiveBayesI, trainInd=x5)
> confuMat(zz)
```

| | predicted | |
|---------|-----------|-----|
| given | BCR/ABL | NEG |
| BCR/ABL | 30 | 7 |
| NEG | 9 | 33 |

```
> names(RObject(RObject(zz)[[1]]$mlans))
[1] "apriori" "tables"  "levels"  "call"
```

Patterns, architecture

- MLEarn is the workhorse method for classification
- parameters
 - formula [familiar but altered]
 - data [either a data.frame or ExpressionSet, more]
 - method/learnerSpec [an adapter to heterogeneous R functions]
 - trainInd/xvalSpec [define the train/validation elements; can include embedded feature selection (see fs.absT)]
 - ... [parameters to the native learning function]

Return values

```
> getClass("classifierOutput")
```

Slots:

| | | |
|--------|--------------|-----------------|
| Name: | testOutcomes | testPredictions |
| Class: | factor | factor |

| | | |
|--------|------------|---------------|
| Name: | testScores | trainOutcomes |
| Class: | ANY | factor |

| | | |
|--------|------------------|-------------|
| Name: | trainPredictions | trainScores |
| Class: | factor | ANY |

| | | |
|--------|-----------|---------|
| Name: | fsHistory | RObject |
| Class: | list | ANY |

| | | |
|--------|------|------------|
| Name: | call | embeddedCV |
| Class: | call | logical |

Recommended computational approach

- Create an `ExpressionSet` with the data of interest
- Choose a train vs test partition or cross-validation setup (which may include feature selection)
- Choose a learner, and identify tuning parameter values
- `fit = MLearn(formula, data, ...`
- `confuMat(fit)` will help appraise accuracy
- elements of `RObject(fit)` may also be helpful

Example: Fisher's linear discriminant analysis and variations

LDA: assuming common covariance Σ , allocate x_i to class 1 when

$$\hat{c}(x_i) = (\hat{\mu}_1 - \hat{\mu}_0)^t \hat{\Sigma}^{-1} (x_i - .5(\hat{\mu}_1 + \hat{\mu}_2)) > \log(\hat{\pi}_0/\hat{\pi}_1)$$

- parametric justification: under separate MVG models for classes with population parameters $\pi_0, \pi_1, \mu_0, \mu_1, \Sigma$, the the LD criterion is given by Bayes' rule (maximizes posterior probability of class membership given data)
- wide-sense justification: seek the vector \mathbf{w} maximizing

$$\frac{\mathbf{w}^t \mathbf{S}_B \mathbf{w}}{\mathbf{w}^t \mathbf{S}_W \mathbf{w}}$$

where \mathbf{S}_B (\mathbf{S}_W) are between (within) class scatter matrices; decision boundary satisfies $c(x) = \mathbf{w}^t x + w_0 = 0$

Variations on LDA

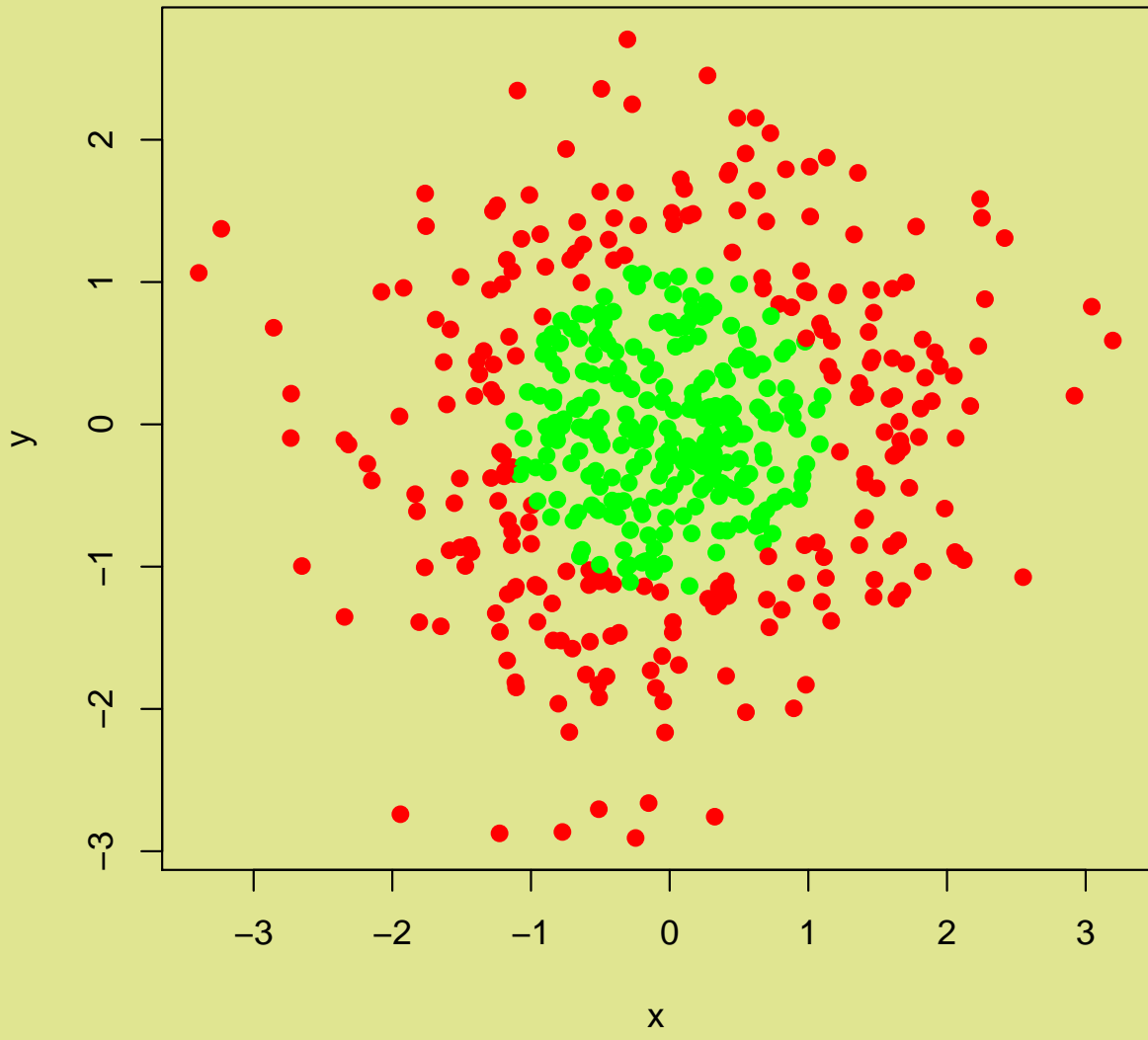
- QDA: allow distinct class-specific covariance matrices
- DLDA: stipulate that Σ is diagonal – avoid estimating $O(p^2)$ covariance parameters
- RDA: (regularized) estimate a tuning parameter α and use $\alpha\Sigma + (1 - \alpha)I$
- SVM: jointly estimate nonlinear transformation of features and discriminator that yields large margin (more later)

Transformations and distances

- the measurement scales for features are not sacrosanct
- however, parameter or rule interpretation on the original scale may be useful for communicating results
- overfitting through transformation is a source of concern
- “the art of data analysis” seems inevitable, counter to the intuition guiding us to “machine learning”

Concentric circular feature configuration

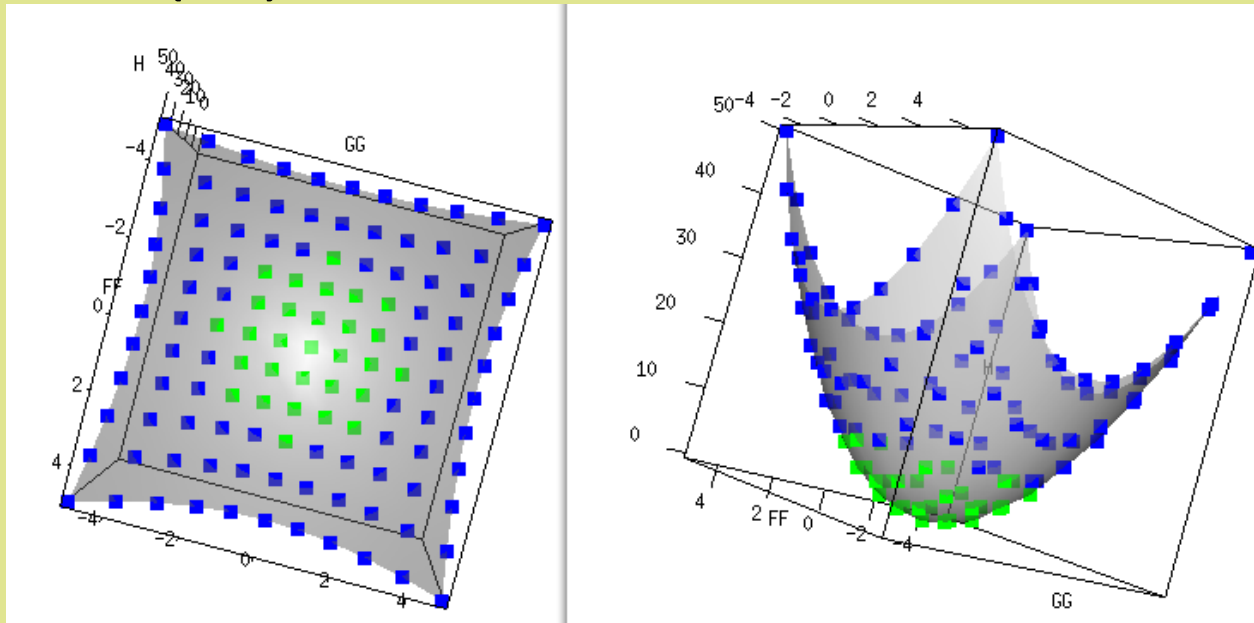
```
> set.seed(1234)
> X = matrix(rnorm(1000),nrow=500)
> from0 = apply(X,1,
+   function(x) sqrt(x[1]^2+x[2]^2))
> cl = 1*(from0 > median(from0))
> colo = ifelse(cl==0,"green", "red")
> plot(X[,1], X[,2], col=colo,
+   pch=19, xlab="x", ylab="y")
> ccdat = data.frame(x=X[,1], y=X[,2],
+   cl=colo)
```



Predicting class using features

- suggest some methods for predicting class on the basis of (x, y)
 - of course if we look at the data and find such a configuration we can measure the radius and state a rule
 - the whole purpose of the “machine learning” focus is to discover and classify using numerical methods (not visualization per se)
- indicate how to develop a prediction rule using familiar linear modeling tools

Embedding two-dimensional features in a three-dimensional space (rgl)



A plane can now be used to discriminate blue and green objects

What procedure should be used to find the plane?

Examples

```
> xv5 = xvalSpec("LOG", 5, balKfold.xvspec(5))
> confuMat(MLearn(cl ~ ., ccdat, ldaI, xv5))
```

```
      predicted
given  green red
green  113 137
red    124 126
```

```
> confuMat(MLearn(cl ~ ., ccdat, knnI(k = 1), xv5))
```

```
      predicted
given  green red
green  242   8
red    10 240
```

```
> confuMat(MLearn(cl ~ poly(x, 2) + poly(y, 2), ccdat, ldaI, xv5,
+ family = binomial))
```

```
      predicted
given  green red
green  250   0
red    62 188
```

```
> confuMat(MLearn(cl ~ poly(x, 2) + poly(y, 2), ccdat, glmI.logistic(0.5),
+ xv5, family = binomial))
```

```
      predicted
given    0   1
green 247   3
red    1 249
```

SVM: an algorithmic variation on LDA

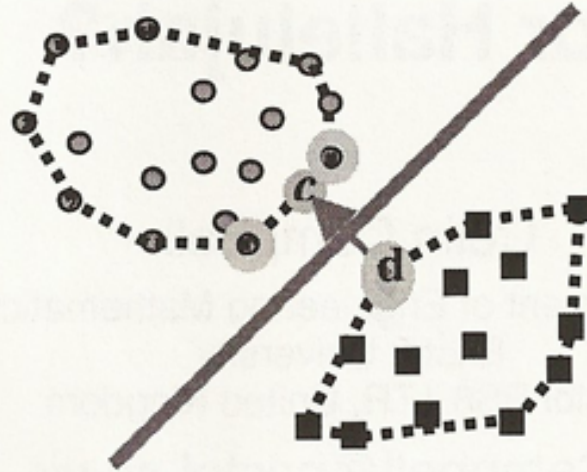
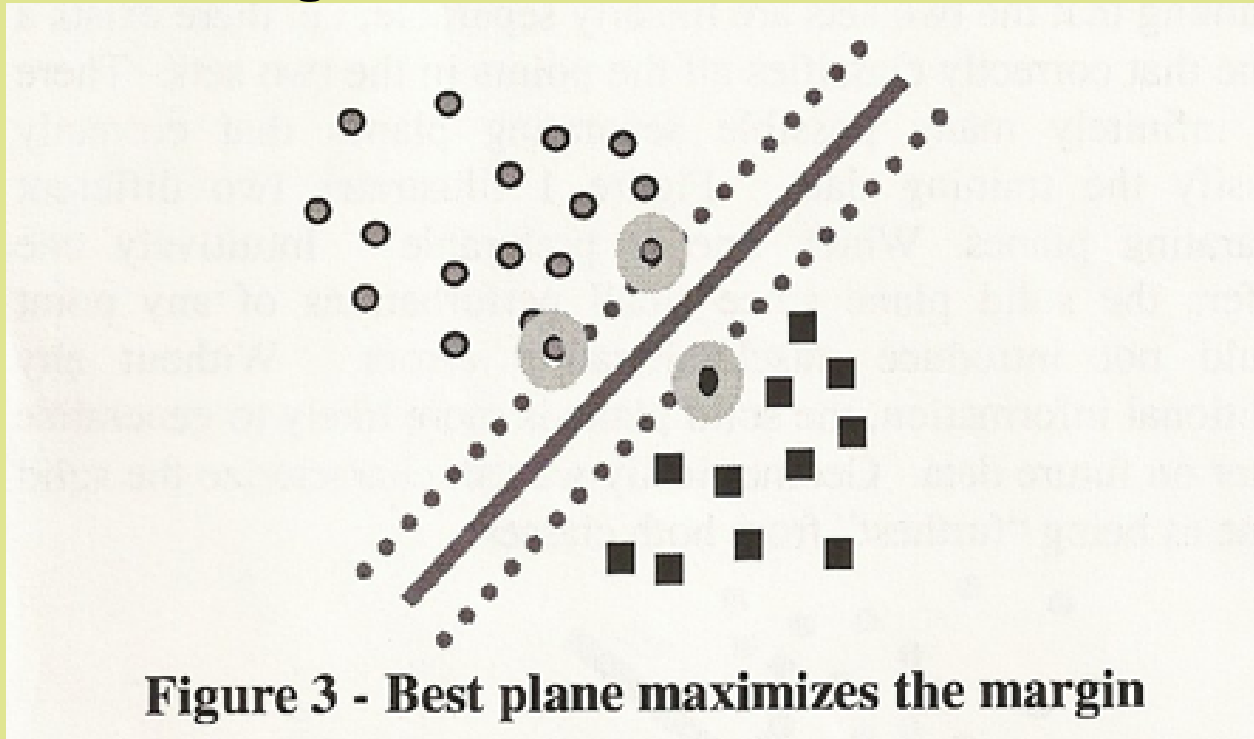


Figure 2 – Best plane bisects closest points in the convex hulls

The closest points in the two convex hulls can be found by solving the following quadratic problem.

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \|c - d\|^2 \\ c = \sum_{y_i \in \text{Class1}} \alpha_i x_i \quad & d = \sum_{y_i \in \text{Class-1}} \alpha_i x_i \\ \text{s.t.} \quad & \sum_{y_i \in \text{Class1}} \alpha_i = 1 \quad \sum_{y_i \in \text{Class-1}} \alpha_i = 1 \\ & \alpha_i \geq 0 \quad i = 1, \dots, m \end{aligned} \quad (1)$$

SVM: an algorithmic variation on LDA



Kristin Bennett et al, SVM: Hype or Hallelujah? ACM SIGKDD Explorations 2000 (citeseer)

SVM for the concentric circles problem

```
> confuMat(MLearn(c1 ~ x + y, ccdat, svmI, xv5))
```

```
      predicted
given  green red
green  241   9
red    6  244
```

```
> confuMat(P2 <- MLearn(c1 ~ x + y, ccdat, svmI, xv5, kernel = "polynomial",
+   degree = 2))
```

```
      predicted
given  green red
green  247   3
red    14  236
```

```
> library(kernlab)
```

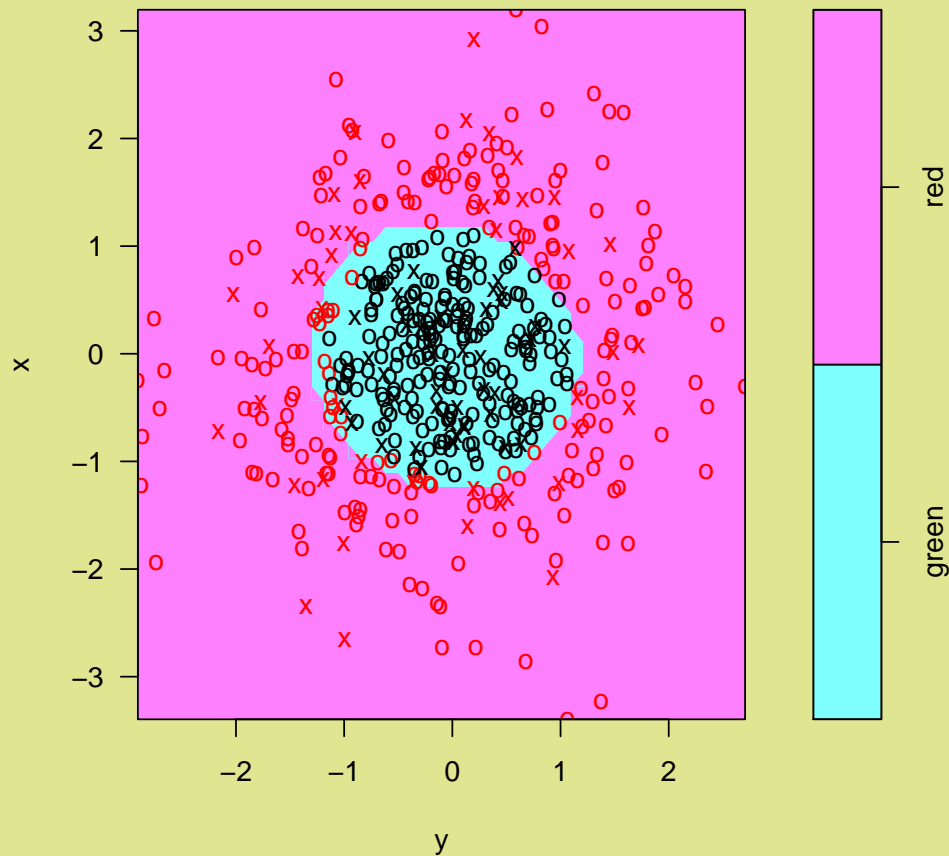
```
> confuMat(MLearn(c1 ~ x + y, ccdat, ksvmI, xv5, kernel = "polydot",
+   kpar = list(degree = 2)))
```

```
      predicted
given  green red
green  246   4
red    2  248
```

Visualizing SVM (slicing also supported)

```
> plot(RObject(RObject(P2)[[1]]$mlans), data = ccdat)
```

SVM classification plot



Summary

- Supervised learning: Data are x_i, c_i , seek the rule $c(x)$ that classifies object bearing features x to its class with high reliability

- LDA: choose w to maximize

$$\frac{w^t S_B w}{w^t S_W w}$$

but the solution will change if we allow polynomial transformations and interactions; same for logistic reg., knn

- the data *per se* do not determine the predictability, but choice of feature space metric/feature transformations play an important role
- Major concern of modern statistical learning theory: How can we use the data to select transformations and model structure and avoid overfitting?

Exploiting sparsity

- Regularized LDA (Guo et al, Biostatistics 2007):

- $\tilde{\Sigma} = \alpha \hat{\Sigma} + (1 - \alpha)I_p$

- choose α to minimize cross-validated misclassification rate (MCR)

- Shrink the centroids: Let $\bar{x}^* = \tilde{\Sigma}^{-1}\bar{x}$

Use

$$\bar{x}^{*'} = \text{sgn}(\bar{x}^*)(|\bar{x}^*| - \Delta)_+$$

- choose α and Δ simultaneously to minimize MCR and to minimize number of contributing genes

```
> rd1 = MLearn(mol.biol ~ ., data = nsALLb, rdacvI, 1:45)
```

```
> confuMat(rd1)
```

| | predicted | |
|---------|-----------|-----|
| given | BCR/ABL | NEG |
| BCR/ABL | 13 | 1 |
| NEG | 2 | 18 |

The returned object

```
> RObject(rd1)

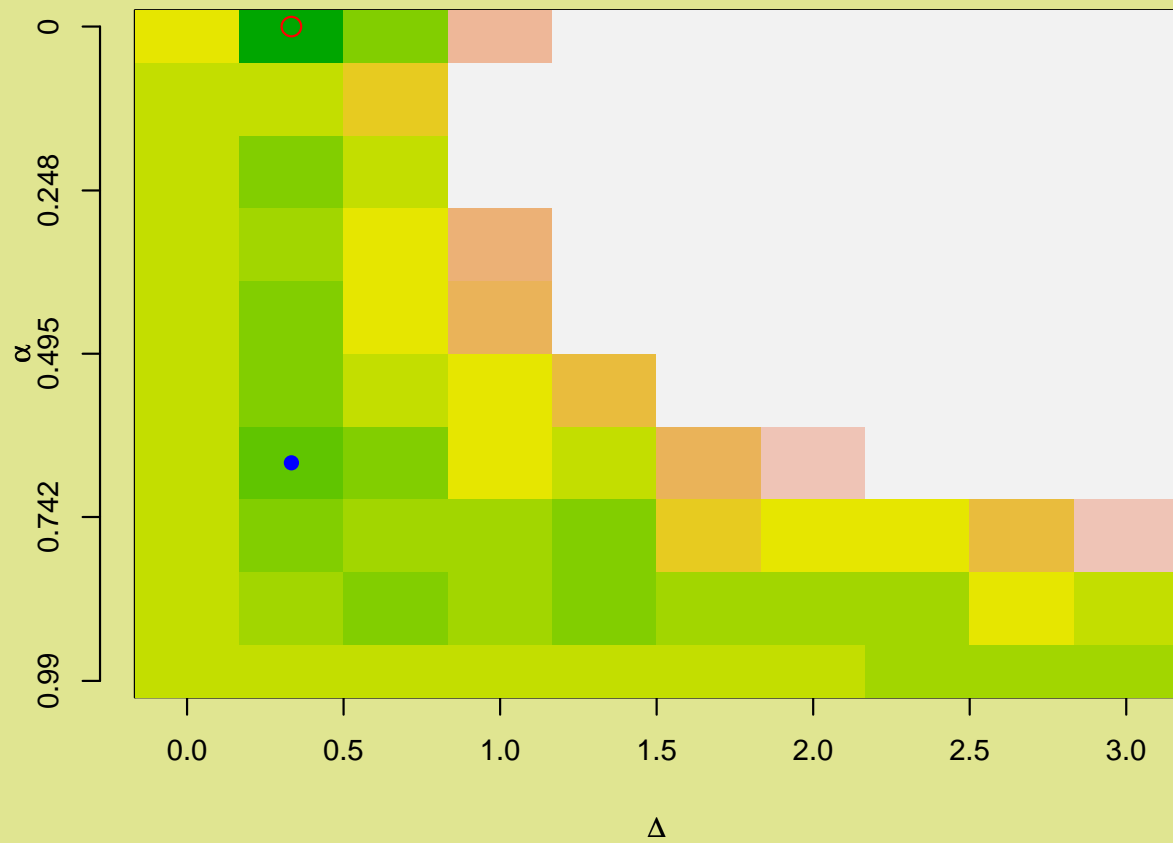
rdacvML S3 instance. components:
[1] "finalFit"      "x"              "resp.num"      "resp.fac"      "featureNames"
[6] "keptFeatures"
---
elements of finalFit:
[1] "alpha"          "delta"          "prior"          "error"
[5] "ngene"          "centroids"      "centroid.overall" "call"
[9] "yhat"           "gene.list"      "reg"
---
the rda.cv result is in the xvalAns attribute of the main object.

> length(RObject(rd1)$keptFeatures)

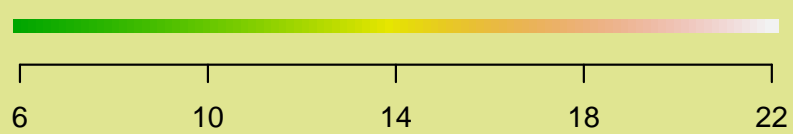
[1] 305
```

```
> zz = plotXvalRDA(rd1, type = "error")
```

Heatmap of 10-fold CV Error



○ Min CV Err.
● 1-SE CV Err.



Color Code

Documentation: Pass-through

randomForestI

[randomForest](#). Note, that to obtain the default performance of randomForestB, you need to set mtry and sampsize parameters to sqrt(number of features) and table([training set response factor]) respectively, as these were not taken to be the function's defaults.

knnI(k=1,l=0)

[knn](#); special support bridge required, defined in MLint

knn.cvI(k=1,l=0)

[knn.cv](#); special support bridge required, defined in MLint. This option uses the embedded leave-one-out cross-validation of `knn.cv`, and thereby achieves high performance. You can have more general cross-validation using `knnI` with an `xvalSpec`, but it will be slower. When using this learner schema, you should use the numerical `trainInd` setting with `1:N` where `N` is the number of samples.

dldaI

[stat.diag.da](#); special support bridge required, defined in MLint

nnetI

[nnet](#)

rpartI

[rpart](#)

ldaI

[lda](#)

svmI

[svm](#)

qdaI

[qda](#)

Summary

- Focused on **how** to do ML with microarrays, reducing user burden
 - Uniform data structure input, model specification
 - Uniform output structure, with natively computed results
- Support flexible cross-validation with embedded feature selection
- Give simple guidelines for interfacing new algorithms
- Note: LDA or DLDA often found competitive with most complex procedures (Hand 2005 StatSci symposium)
- interactions, variable transformations frequently ignored, or obtained implicitly (kernel selection)
- aim of Bioconductor: make it easy to do what you want in a reproducible comparative framework