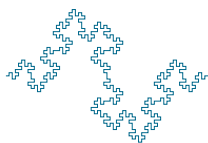


# High-Throughput Sequencing Course

## Supervised Learning

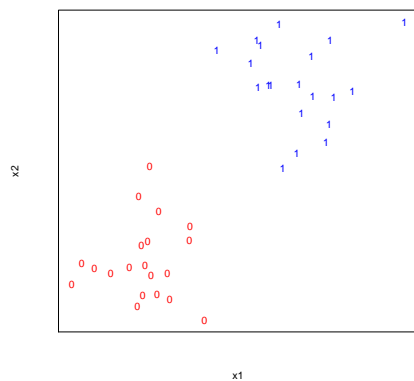
Biostatistics and Bioinformatics



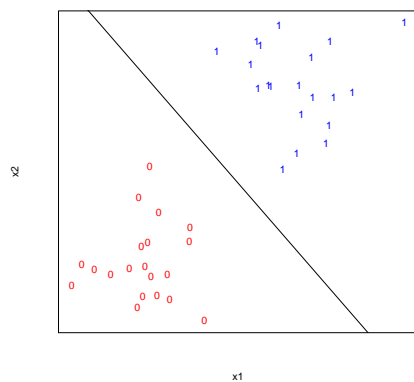
Summer 2018



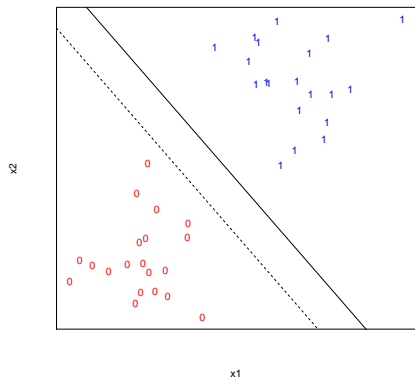
## CLASSIFICATION PROBLEM



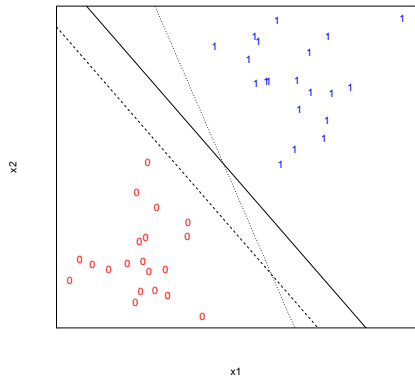
## CLEAR-CUT CASE



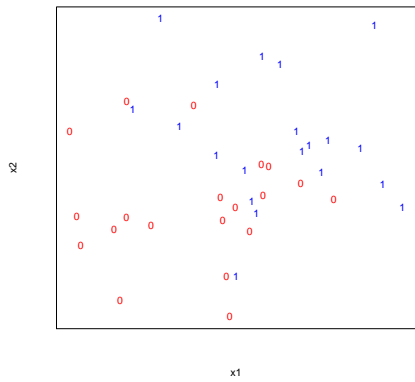
CLEAR-CUT CASE?



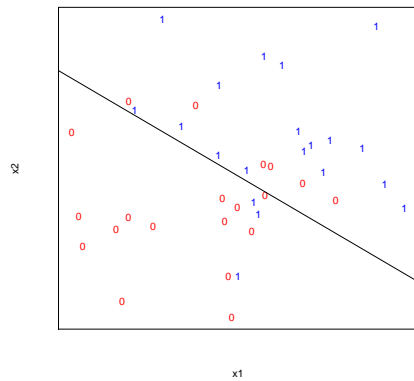
CLEAR-CUT CASE??



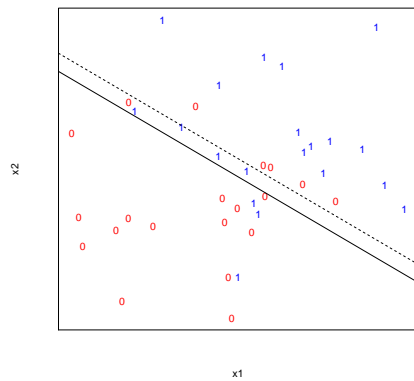
LESS CLEAR-CUT CASE



## LESS CLEAR-CUT CASE

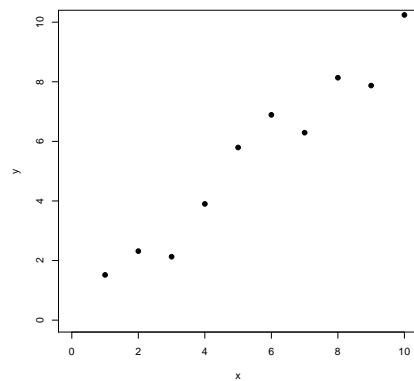


## LESS CLEAR-CUT CASE



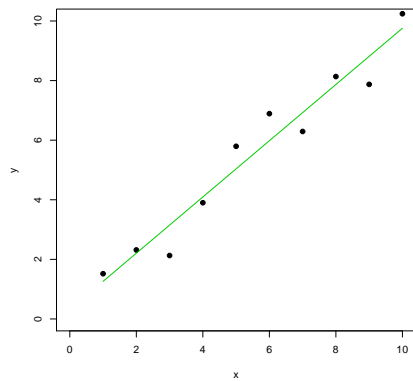
## REGRESSION PROBLEM

```
set.seed(10) x <- 1:10 y = x + rnorm(10, 0.5)
par(mfrow = c(1, 1), bg = "white") plot(x, y, xlim = c(0, 10), ylim = c(0, 10), pch = 19)
```

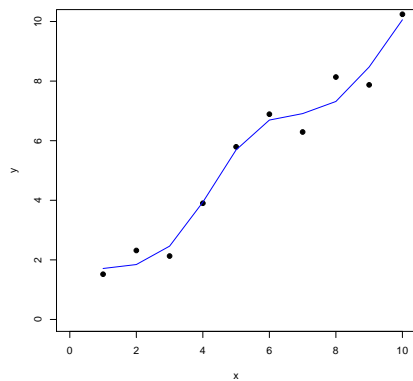


## LINEAR REGRESSION (LIN)

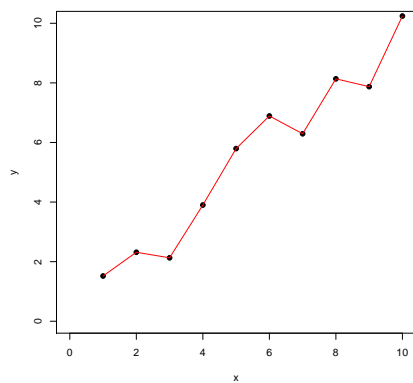
```
par(mfrow = c(1, 1), bg = "white") plot(x, y, xlim = c(0, 10), ylim = c(0, 10), pch = 19) modlm =  
lm(y ~ x) lines(x, predict(modlm), col = 3)
```



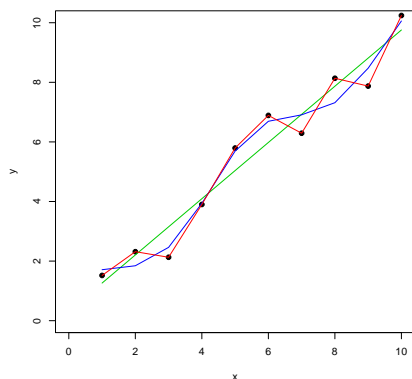
## SPLINE REGRESSION (SPL)



## CONNECT THE DOTS (CTD)



## WHICH APPROACH?



## SUPERVISED LEARNING (CLASSIFICATION)

- ▶ Goal: Predict a binary outcome ( $Y$ ) on the basis of baseline information ( $X$ )
- ▶  $Y$  assumes the value 0 or 1 (e.g., control vs case, or AML vs ALL)
- ▶  $X$  could be single variable or be a vector of multiple variables
- ▶ Example: Can you predict  $Y$  on the basis of two genes say  $X_1$  and  $X_2$
- ▶ Note that a goal is to build a machine that will take on two values  $X_1$  and  $X_2$  and return a 0 or a 1
- ▶ You can denote this machine as a function  $g(x_1, x_2)$

## CLASSIFIER

- ▶ We will denote the predictor or classifier by  $g(x)$
- ▶  $x = (x_1, x_2)$  is the vector of gene expressions for genes 1 and 2
- ▶ Based on  $x$ , the classifier  $g$  makes a prediction for the outcome
- ▶ Note that  $g(x) = 0$  or  $g(x) = 1$
- ▶ The prediction is *correct* if  $Y = 1$  and  $g(x_1, x_2) = 1$ , or  $Y = 0$  and  $g(x_1, x_2) = 0$
- ▶ The prediction is *wrong* if  $Y = 0$  and  $g(x_1, x_2) = 1$ , or  $Y = 1$  and  $g(x_1, x_2) = 0$

## PREDICTION ASSESSMENT

	$g(x_1, x_2) = 0$	$g(x_1, x_2) = 1$
$Y = 0$	True-Negative	False-Positive
$Y = 1$	False-Negative	True-Positive

## STEPS TO CONSTRUCT A CLASSIFIER

- ▶ Collect a random data set of size  $n$  to build (train) a classifier
- ▶ This is called the training data
- ▶ On the basis of these data, construct the classifier  $g_n$
- ▶ It is subscripted by  $n$  to emphasize that it is trained on the basis of the training data
- ▶ Note that the final performance of  $g_n$  is *not* be judged on the basis of the training data
- ▶ It is to be judged on the basis of its performance on *future* data
- ▶ Called testing data

## STEPS IN NOTATION

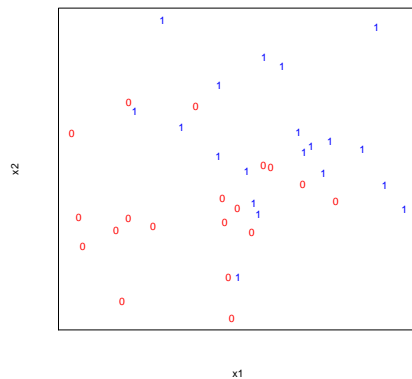
- ▶ Collect the training data  $(X_1, Y_1), \dots, (X_n, Y_n)$
- ▶ Construct a classifier  $g_n$  on the basis of the training data
- ▶ Apply  $g_n$  to a new data set  $X_1^*, \dots, X_k^*$  to get
- ▶  $k$  predictions:  $\hat{Y}_1^*, \dots, \hat{Y}_k^*$
- ▶ Compare the predictions to the observed outcomes  $Y_1^*, \dots, Y_k^*$
- ▶ Note that at the testing stage, you are blinded to the  $Y_k^*$

## $k$ -NEAREST NEIGHBORHOOD (NON-PARAMETRIC)

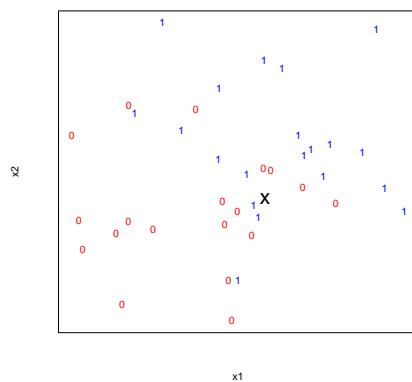
- ▶ Generally, non-parametric methods (e.g.,  $k$ -NN) are preferred
- ▶ These do not make strong assumptions on the specific shape of the underlying relationship, if there is one, between  $X$  and  $Y$
- ▶ For each  $x$  (point on the scatter plot), identify the  $k$  nearest neighbors
- ▶ Among the  $k$  neighbors, count the number of responders (say  $r_x$ )
- ▶ Set

$$\hat{\eta}(x) = \frac{r_x}{k}$$

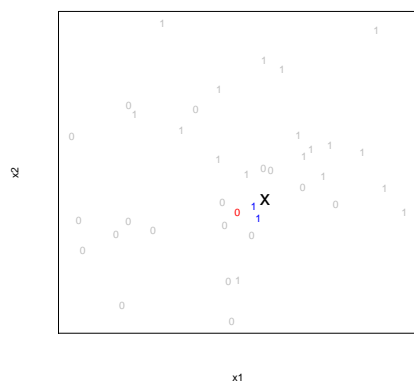
## K-NEAREST NEIGHBORHOOD (TRAINING DATA)



## K-NEAREST NEIGHBORHOOD (PREDICT NEW SAMPLE)

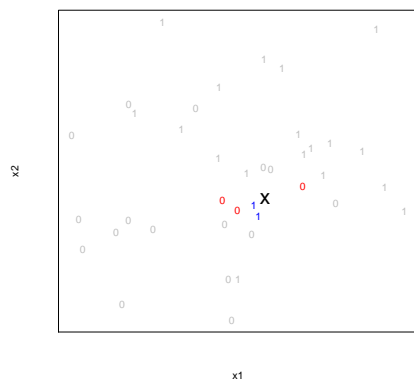


### 3-NEAREST NEIGHBORHOOD



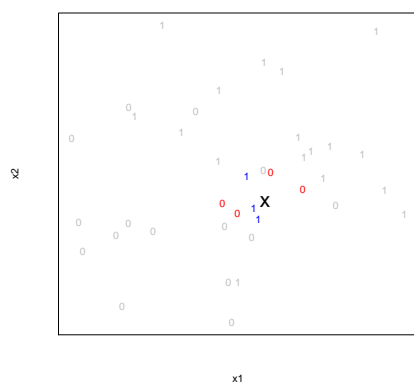
The prediction based on 3 nearest neighbors is  $\hat{Y}^* = 1$

### 5-NEAREST NEIGHBORHOOD



The prediction based on 5 nearest neighbors is  $\hat{Y}^* = 0$

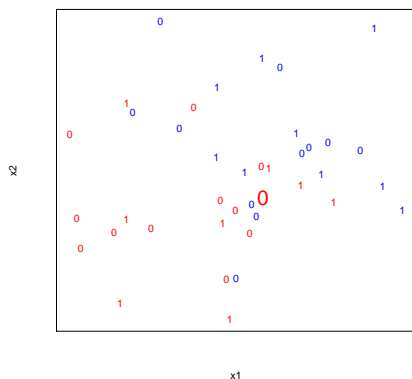
### 7-NEAREST NEIGHBORHOOD



The prediction based on 7 nearest neighbors is  $\hat{Y}^* = 0$



## PREDICTION VS TRUE STATE



Question: What should the parameter  $k$  be?

## MEAN REGRESSION MODEL

- $E(Y)$  is the *unconditional* (on  $X$ ) mean of  $Y$ .
- Model the mean relationship between  $Y$  and  $X$

$$\eta(x) = E(Y|X = x)$$

- $\eta(x)$  is the *conditional* (on  $X$ ) mean of  $Y$  given that  $X$  has realized the value  $x$ .

## BAYES CLASSIFIER

$$g(X) = \begin{cases} 1 & \text{if } \eta(X) \geq \frac{1}{2}, \\ 0 & \text{if } \eta(X) < \frac{1}{2} \end{cases}$$

- This classifier is "optimal" in the sense that there is no better classifier with respect to minimizing the error ( $P(g(X) \neq Y)$ ).
- Suppose that  $g^*$  is another classifier. Then

$$P(g(X) \neq Y) \leq P(g^*(X) \neq Y)$$

- Note that the optimality concerns  $\eta(x)$  and not  $\hat{\eta}(x)$ .

## LOGISTIC REGRESSION (PARAMETRIC)

- ▶ Most commonly used method for modeling the relationship between a binary response and a set of co-variables

$$\text{logit}(\eta(x)) = \beta_0 + \beta_1 x_1 + \beta_2 x_2,$$

where

$$\text{logit}(\eta(x)) = \log\left(\frac{p}{1-p}\right),$$

for  $p \in (0, 1)$  is called the "logit" function.

## ESTIMATING $\eta(x)$

- ▶ Estimate the model parameters ( $\beta_0, \beta_1$  and  $\beta_2$ ) using maximum-likelihood estimation to get  $\hat{\beta}_0, \hat{\beta}_1$  and  $\hat{\beta}_2$
- ▶ For the logistic model

$$\hat{\eta}(x) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2)}$$

## OTHER CLASSIFICATION METHODS

- ▶ Fisher's Linear Discriminant
- ▶ Support Vector Machines (SVM)
- ▶ Classification and Regression Trees (CART)
- ▶ Random Forests (aggregated trees)
- ▶ Methods for "Deep" learning

## BIAS VERSUS VARIANCE

- ▶ A very important principle in statistical modeling is the so called *bias-variance tradeoff*
- ▶ The bias of  $\hat{\eta}(x)$  is

$$b(x) = \hat{\eta}(x) - \eta(x)$$

- ▶ The variance of  $\hat{\eta}(x)$  is

$$v(x) = E(\hat{\eta}(x) - \eta(x))^2$$

- ▶ The bias-variance tradeoff implies that both cannot be minimized simultaneously
- ▶ For example for the  $k$ -NN method increasing  $k$  increases bias while decreasing variance

## TRAINING AND TESTING

- ▶ In practice, the model is first estimated (trained) using an initial set of data
- ▶ This data set is usually called the "training" data
- ▶ Once the model is trained, then it is applied to an "independent" set of data
- ▶ This data set is usually called the "testing" (or validation) data set

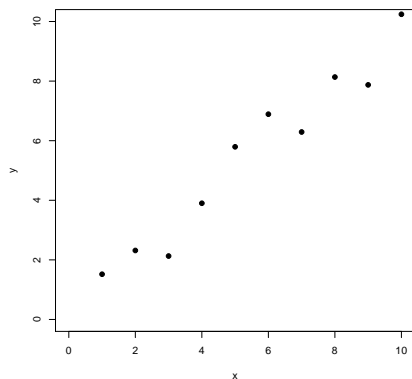
## PARSIMONY

- ▶ The model should be parsimonious (less is more)
- ▶ Including too many noisy/unimportant features often degrades the performance of the classifier.
- ▶ Including highly dependent induces problems (e.g., multi-collinearity from simple linear regression).
- ▶ Additional complication: It is not practically/computationally feasible to include tens of thousands of features in the model.

## OVERFITTING

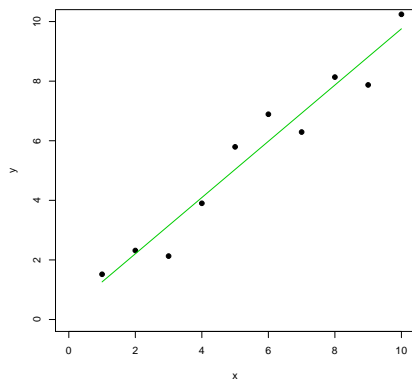
- ▶ Too many parameters compared to the number of data points in the training set
- ▶ A complicated model will fit the training set well
- ▶ It will however perform poorly for an independent set.

## OVERFITTING

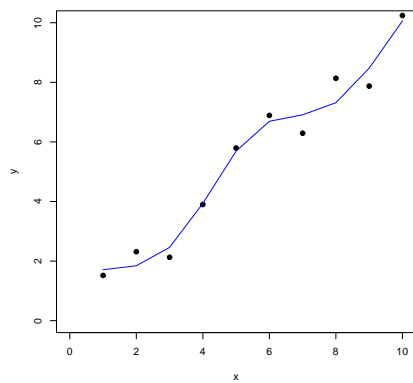


## LINEAR REGRESSION (LIN)

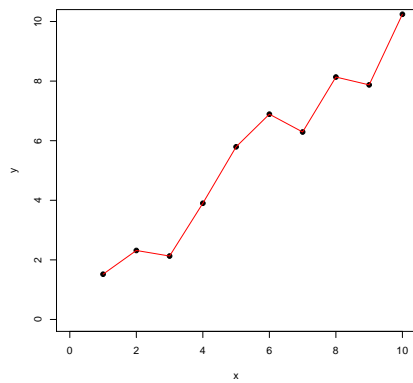
```
par(mfrow = c(1, 1), bg = "white") plot(x, y, xlim = c(0, 10), ylim = c(0, 10), pch = 19) modlm =  
lm(y ~ x) lines(x, predict(modlm), col = 3)
```



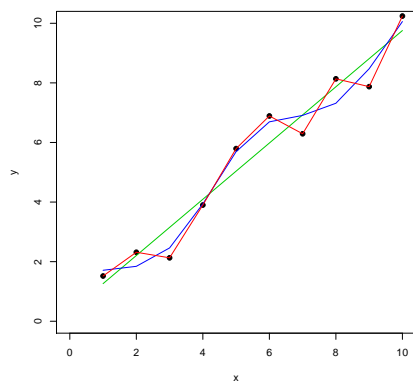
## SPLINE REGRESSION (SPL)



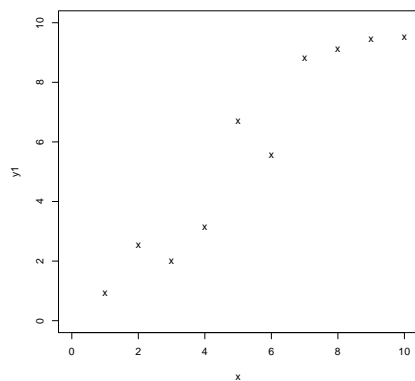
## CONNECT THE DOTS (CTD)



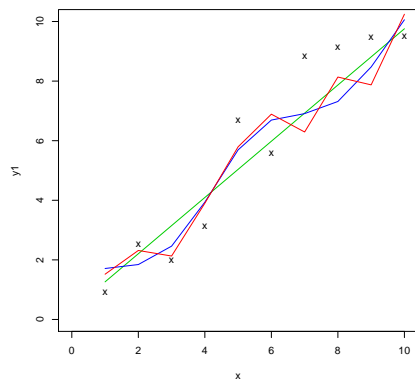
RSS: 4.1 (LIN) VS 1.9 (SPL) VS 0 (CTD)



## NEW DATA SET



RSS: 11 (LIN) vs 12.4 (SPL) vs 14 (CTD)



## TWO CHALLENGES IN BUILDING A CLASSIFIER

### 1. Feature Selection:

- ▶ It is neither feasible nor provident to build a classifier based on all available variables
- ▶ A subset of the variables has to be selected to build the model
- ▶ This is also called feature extraction

### 2. Tuning Parameter Selection:

- ▶ Statistical methods may have one or more parameters that have to be set
- ▶ For example when using  $k$ -NN, one has to decide what  $k$  should be (e.g., 1, 3 or 5 or how about 8)?
- ▶ Choosing the defaults set by the software is inappropriate
- ▶ The feature selection method could also have tuning parameters that have to be set (e.g., the number of features to be selected)
- ▶ The performance of the method could be highly sensitive to the choice of these parameters

## FEATURE SELECTION

- ▶ Reasonable Feature Selection is *critical* if not the most important component of model building.
- ▶ You cannot expect to build a good model if you select poor features.
- ▶ This is also called Feature Extraction
- ▶ We will talk about a few approaches that have been used in the literature.

## FEATURE SELECTION (RANKED BASED ON TEST-STATISTIC)

- ▶ Compute the two-sample t-test for all  $m$  features (based on the training set)
- ▶ Identify the top say 10 or 15 features (e.g, ranked based on the absolute value of the test statistic).
- ▶ Build a model on these "top" features (based on the training set)
- ▶ Alternatively, you could select all features for which the  $P$ -value is less than a certain threshold (say 0.001).
- ▶ You can also use the Wilcoxon rank sum statistic to protect against choosing features with outliers.

## FEATURE SELECTION (ORDINATION METHODS)

- ▶ A standard approach for reducing the dimension in the microarray setting is the method of Principal Components (PCs)
- ▶ The PCs are combinations of the original variables (gene expressions) that have maximum variability
- ▶ They are also constructed as to be uncorrelated with another
- ▶ This attempts to address the issue of high dimension and multi-collinearity simultaneously.
- ▶ One can use the principal components (say the first two or three) as the features
- ▶ Alternatively, one can first reduce the dimension by using the two-sample test-statistic approach and then get the PCs

## TUNING

- ▶ You cannot expect to be able to build a model using default values provided by the software package.
- ▶ If you use  $k$ -NN you need to decide which  $k$  (e.g., 3 or 5 or 7) you want to use
- ▶ If you use the simple feature selection method you need to determine how many "top" features you want to use
- ▶ If you are doing PC dimension reduction, you need to determine how many PCs you want to use.
- ▶ In some books and articles, "tuning" only refers to the choice of the model parameter (e.g.,  $k$  in  $k$ -NN)
- ▶ Must take a broader perspective as the choices in the FS part also affect the results.

## VALIDATION

- ▶ Split the data into a training and a mutually exclusive testing set
- ▶ Build the model (including feature selection, tuning) on the *training* set
- ▶ Evaluate the performance of the model on the *testing set*
- ▶ IMPORTANT: The model is built based on the *training* set. The *testing* set should not contribute *any* information.
- ▶ Violating this principle will invariably result in bias

## ERROR SUBSTITUTION VALIDATION

- ▶ Error Substitution Validation: The testing set is empty.
- ▶ Test the model you just built on the *training* set
- ▶ This approach cannot be recommended under any circumstance.
- ▶ Analogy: Assess the fit of the linear model by plotting the fitted (from the data) to the observed data.
- ▶ A bona-fide testing set is required.
- ▶ Will demonstrate how this can lead to noise discovery



## HOLD-OUT METHOD

- ▶ Split the data into two parts
- ▶ Keep the testing set locked up
- ▶ Better yet, ask an "honest" broker to keep it from you until you are ready to test the model
- ▶ This approach is reasonable if you have a large number of cases
- ▶ It may be problematic if the outcomes are sparse

## $k$ -FOLD CROSS-VALIDATION

- ▶ Many microarray experiments are from smaller (e.g., pilot) studies
- ▶ It is not impossible to get reasonably size training and testing sets this cases
- ▶ A reasonable approach to get around this is  $k$ -fold cross-validation (CV)
- ▶ Randomly split cases into  $k$  (nearly) equally sized subsets (folds).
- ▶ At each step take of these  $k$  portions as the *testing* set and construct the *training* set based on the other  $k - 1$  portions
- ▶ Special case is Leave-One-Out CV (LOOCV) where  $k = n$
- ▶ For really small data sets, LOOCV is often the best (most practical) choice.

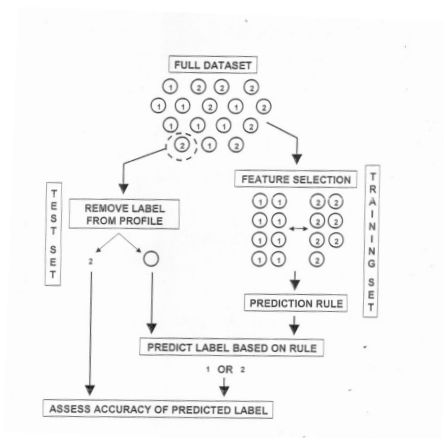
## NAIVE CROSS-VALIDATION

- ▶ Naive Validation: Do the feature selection once based on all  $n$  cases
- ▶ In each CV step use the same set of features.
- ▶ This will invariably make the results look better than they really are
- ▶ It should be avoided unless one feels *very* certain about the features (say biologically relevant gathered *a priori*)

## PROPER CROSS-VALIDATION

- ▶ Choose the first fold and set it aside the other  $k - 1$  folds
- ▶ Carry out Feature Selection on the other  $k - 1$  folds
- ▶ Train the model based the top features on the  $k - 1$  folds
- ▶ Test the model on the first fold left out
- ▶ Repeat the above for the second fold (set aside the second fold, leave in the first and the next  $k - 2$  folds).

## IMPORTANT ILLUSTRATION (FIG 8.5) FROM SIMON ET AL.



## SIMULATE DATA FOR $k$ -NN PREDICTION

- ▶ Simulate expression from 1000 genes for 40 patients. Let the first 20 be responders and the remaining 20 be non-responders

```
set.seed(123)
n = 20
m = 1000
EXPRS = matrix(rnorm(2 * n * m), 2 * n, m)
rownames(EXPRS) = paste("pt", 1:(2 * n), sep = "")
colnames(EXPRS) = paste("g", 1:m, sep = "")
grp = rep(0:1, c(n, n))
```

- ▶ Pick the top 10 features based on the two-sample  $t$ -test

```
stats = abs(rowttests(t(EXPRS), factor(grp)))$statistic
ii = order(-stats)
```

- ▶ Filter out all genes except the top 10

```
TOPEXPRS = EXPRS[, ii[1:10]]
```

## ERROR RESUBSTITUTION AND NAIVE CV

- Error resubstitution (Training and Testing set are the same)

```
mod0 = knn(train = TOPEXPRS, test = TOPEXPRS, cl = grp, k = 3)
table(mod0, grp)

##      grp
## mod0  0  1
##      0 17  0
##      1  3 20
```

- Cross-validated predictions (the features selection is not part of the CV process)

```
mod1 = knn.cv(TOPEXPRS, grp, k = 3)
table(mod1, grp)

##      grp
## mod1  0  1
##      0 16  0
##      1  4 20
```

- Note that in both examples, TOPEXPR not EXP is used.

## R FUNCTION TO IMPLEMENT PROPER CV BASED ON $k$ -NN

```
top.features <- function(EXP, resp, test, fsum) {
  top.features.i <- function(i, EXP, resp, test, fsum) {
    stats <- abs(mt.teststat(EXP[, -i], resp[-i], test = test))
    ii <- order(-stats)[1:fsum]
    rownames(EXP)[ii]
  }
  sapply(1:ncol(EXP), top.features.i, EXP = EXP, resp = resp, test = test,
        fsum = fsum)
}

# This function evaluates the knn

knn.loocv <- function(EXP, resp, test, k, fsum, tabulate = FALSE, permute = FALSE) {
  if (permute)
    resp = sample(resp)
  topfeat = top.features(EXP, resp, test, fsum)
  pids = rownames(EXP)
  EXP = t(EXP)
  colnames(EXP) = as.character(pids)
  knn.loocv.i = function(i, EXP, resp, k, topfeat) {
    ii = topfeat[, i]
    mod = knn(train = EXP[-i, ii], test = EXP[i, ii], cl = resp[-i], k = k)[1]
  }
  out = sapply(1:nrow(EXP), knn.loocv.i, EXP = EXP, resp = resp, k = k, topfeat = topfeat)
  if (tabulate)
    out = ftable(pred = out, obs = resp)
  return(out)
}
```

## PROPER CROSS-VALIDATION

- Finally, we conduct proper cross-validation using the previous R function
- At each iteration, the top 10 features are selected based on the data from the  $n - 1$  samples in the training set

```
knn.loocv(t(EXPRS), as.integer(grp), "t.equalvar", 3, 10, TRUE)

##      obs  0  1
## pred
##  0       7  7
##  1      13 13
```

- Note that EXPRS not TOPEXPR is used.
- The classification rate is 50% (as expected)

## NAIVE LOOCV: QUANTITATIVE TRAIT

- ▶ Repeat the last experiment with a noisy quantitative outcome
- ▶ First simulate a data matrix of dimension  $n = 50$  (patients) and  $m$  (genes)
- ▶ Next draw the outcome for  $n = 50$  patients from a standard normal distribution independent of the data matrix
- ▶ There is no relationship between the expressions and the outcome (by design)
- ▶ We consider  $m = 45$  and  $m = 50000$
- ▶ We conduct Naive LOOCV using the top 10 features

## NAIVE LOOCV: QUANTITATIVE TRAIT

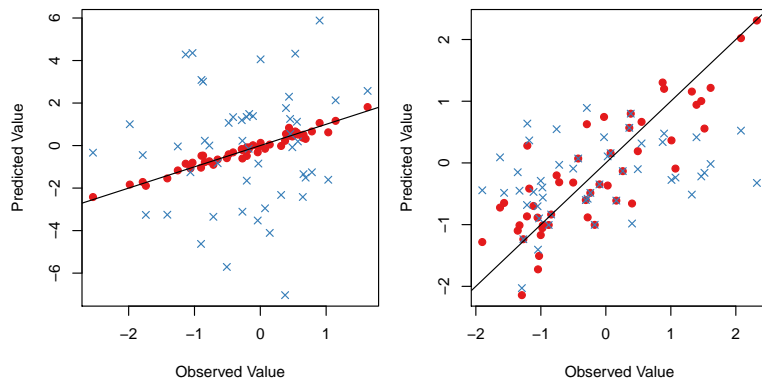


Figure taken from Owzar *et al*; *Clin Transl Sci* 2011.

## TRAINING, VALIDATION AND TESTING APPROACH

- ▶ Before you test the model, you must freeze it
- ▶ You may want to split the Training set further into a Training and Validation set
- ▶ Use the Validation set to "tune" the model.

## FINAL REMARKS

- ▶ It is OK to try different methods (other classifiers, feature selection or tuning methods)
- ▶ Keep track of what you have done and report it (brief description in the paper and details in supplementary material)
- ▶ Be careful if you have too few responders
- ▶ You could have a model that will classify most patients as a non-responder.
- ▶ In this case a 00 ( $Y = 0$  and  $g(X) = 0$ ) may not be bona-fide true-negative
- ▶ The gold-standard for model validation, is to follow up the cross-validation by permutation resampling
- ▶ The R function provided can be used for this purpose

## PRE-PROCESSING CHALLENGE

- ▶ The  $X$  profiles from the testing set need to be "compatible" to those used to train the model
- ▶ In classical experiments with a few biomarkers, the labs had internal controls to ensure that the measurements were properly normalized
- ▶ For RNA-Seq data, you observe counts (not expressions)
- ▶ Number of reads mapped to genes are *not* comparable
- ▶ Why?
- ▶ The current "state" of the art is to "normalize" the counts into expressions
- ▶ This is a practical but not rigorous solution
- ▶ One has to up the ante if the classifier is to be used for important decision (e.g., treating a patient with a toxic but potentially effective drug)

## ON DATA AND ANSWERS

"The data may not contain the answer. The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data."

John Wilder Tukey