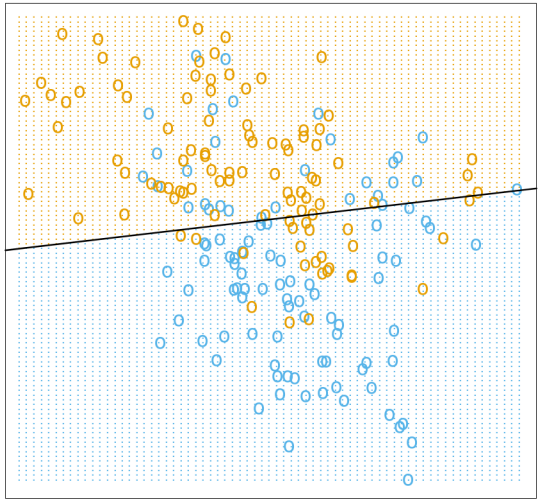
Prediction analysis & Machine learning

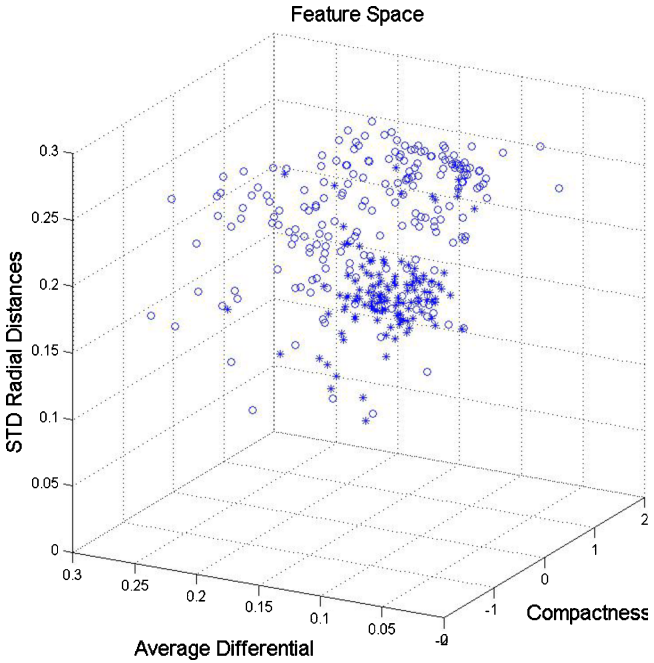
Cory Giles - January 17, 2013 - R/BioC for Microarrays



What is prediction analysis?

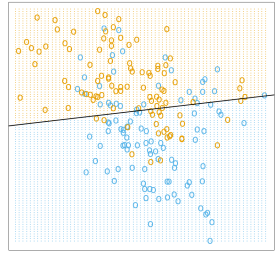
- Use gene expression and/or other data to predict **categorical** or **continuous** variables.
- **For example:**
    - Tumor or normal?
    - Cell type? Tumor subtype?
    - Drug response?
    - Blood pressure?
- *Classification* vs *regression*
- Prediction analysis = *machine learning* applied to biological problems

Feature space



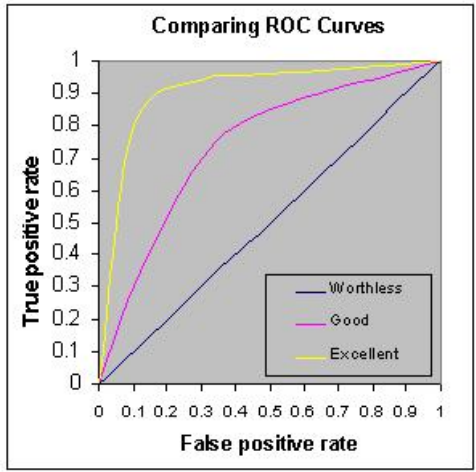
Goal of classification

Classification: Find a "decision boundary" or "hyperplane" that *optimally* separates categories.



- Linear & logistic regression
- Nearest neighbors
- Support vector machine (SVM)
- And many more...

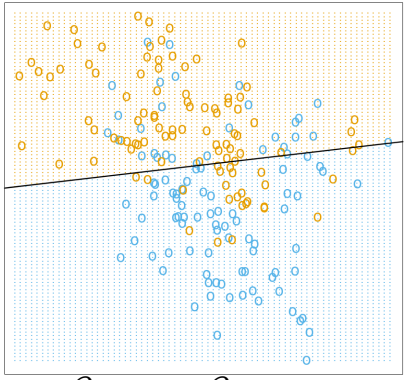
The optimal classification boundary depends on misclassification costs



- ALWAYS a tradeoff between false positives and false negatives (except in very trivial problems)

Linear regression

.



$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i$$

The model consists of $\beta_0 \ldots \beta_i$, which must be learned from the data.

Linear regression

How to estimate $\beta_0 \ldots \beta_i$?

Two common algorithms:

- Ordinary least squares (OLS) - exact answers, **but** variables must be uncorrelated (not a good assumption for gene expression!)
- Gradient descent (approximate)

Linear regression optimizes for the minimization of the **residual sum of squares** (RSS):

- $sum(Y_{pred} - Y_{actual})^2$

Linear regression in R

```
> library(ALL); data(ALL)
> age <- pData(ALL)$age
> m <- exprs(ALL)[,!is.na(age)]
> age <- age[!is.na(age)]
> model <- lm(age ~ m["1000_at",] + \
    m["1001_at",] + m["1002_f_at",])
> model$coef
(Intercept)   m["1000_at", ]
 5.152022          2.984742
m["1001_at", ] m["1002_f_at", ]
 3.970653         -3.897614
```

Linear models for classification

- What about categorical variables? (rather than continuous)
- One *could* simply vary a threshold on the linear regression
- Linear regression optimizes for minimized RSS, *not* classification accuracy
- Also, it would be nice to have a *posterior probability* estimate for $p(Y|X)$

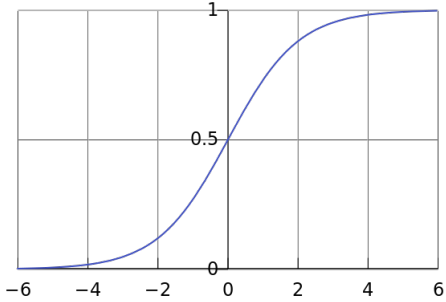
Logistic Regression

Fits the logistic function:

$$\pi = \frac{e^z}{1+e^z}$$

where

$$z = \beta_0 + x_1\beta_1 + x_2\beta_2$$



LR is a "generalized linear model".

Logistic Regression

Compared with linear regression:

- Outputs predictions for Y in [0,1], rather than [-inf,inf]
- Maximizes
- Estimates $p(Y|X)$
- No closed-form solution, model must be fit with iterative methods
- **Model takes longer to converge: need ~10 samples per variable**

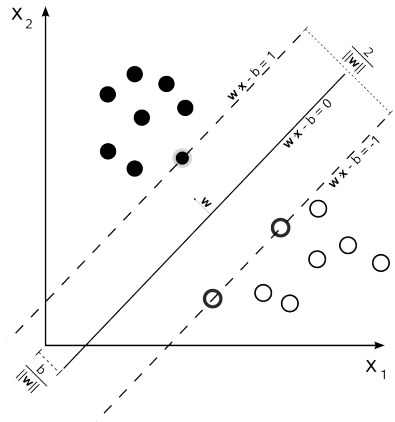
- => do *not* use logistic regression with all genes for microarray prediction analysis, must first select most informative variables

Logistic Regression in R

```
> library(ALL); data(ALL)
> cell.type <- as.factor(substr(pData(ALL)$BT,1,1))
> m <- exprs(ALL)
> model <- glm(age ~ m["1000_at",] + \
    m["1001_at",] + m["1002_f_at",], \
    family=binomial("logit"))
> model$coef
```

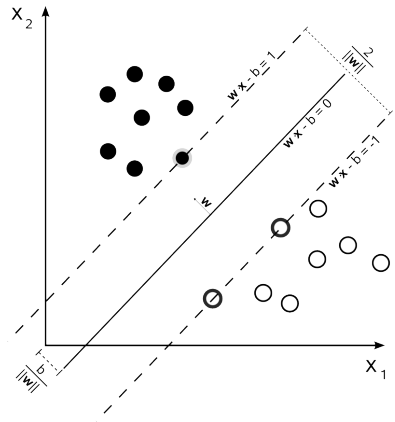
Support vector machine (SVM)

- Invented by Vladimir Vapnik and Corinna Cortes in 1995
- One of the most flexible and commonly-used learning algorithms
- Goal of model: find the **maximum margin hyperplane** separating classes


X2
X1
w x - b = 1
w x - b = 0
w x - b = -1
w
2
||w||
b
||w||

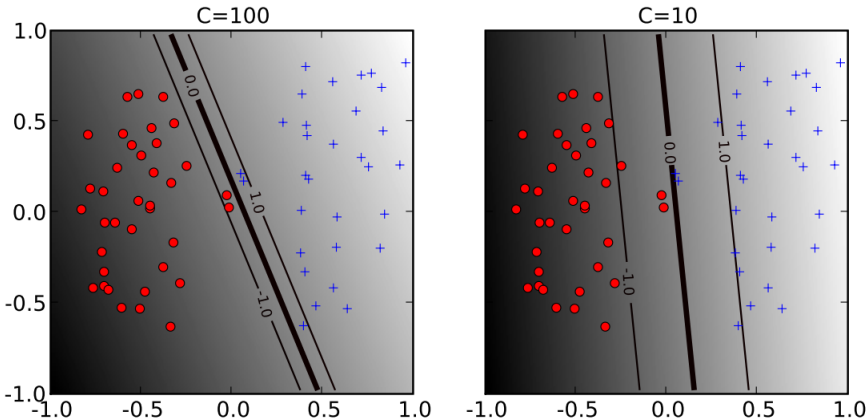
Support vector machine (SVM)

- Also have probabilistic extensions. Intuitively, the further from the decision boundary, the more extreme the probability.
- Only directly applicable to binary classification; multi-class classification is done by one-vs-all (OVA) or one-vs-rest (OVR) strategies


X2
X1
w x - b = 1
w x - b = 0
w x - b = -1
w
2
||w||
b
||w||

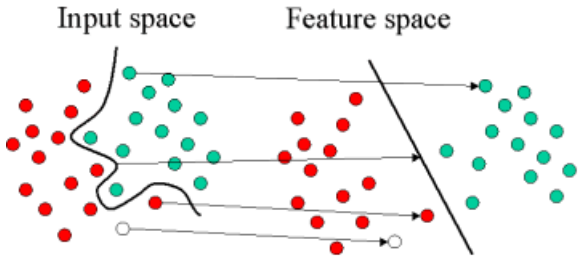
Soft-margin classification

- Not all problems are **linearly separable** (e.g., XOR)
- Soft-margin hyperparameter "C" affects how many **support vectors** will be used



SVM kernel allows nonlinear classification

- A **kernel** maps points from one feature space to another
- The **kernel trick** means that, with certain kernels (those used by SVM), inner products between samples can be calculated without explicitly performing the mapping
- Thus some of the "target" spaces can even be infinite-dimensional, because they are never explicitly calculated!


Input space
Feature space

SVM hyperparameters

A hyperparameter is a user-specified parameter to the model training function. These greatly affect performance, but are not learned like ordinary parameters!

For SVM:

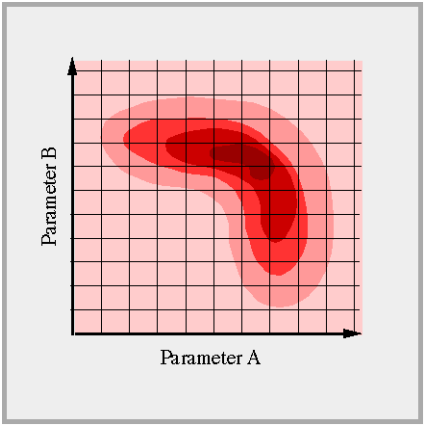
- SVM kernel (linear, polynomial, radial basis function, gaussian, and more)
- **Cost regularization hyperparameter ("C")**
    - Higher cost => higher penalty for misclassified instances => higher probability of overfitting

- Sometimes, instead of C, lambda (= 1/C) is used

- Kernel-specific parameters. gamma (Gaussian), degree (polynomial), etc.

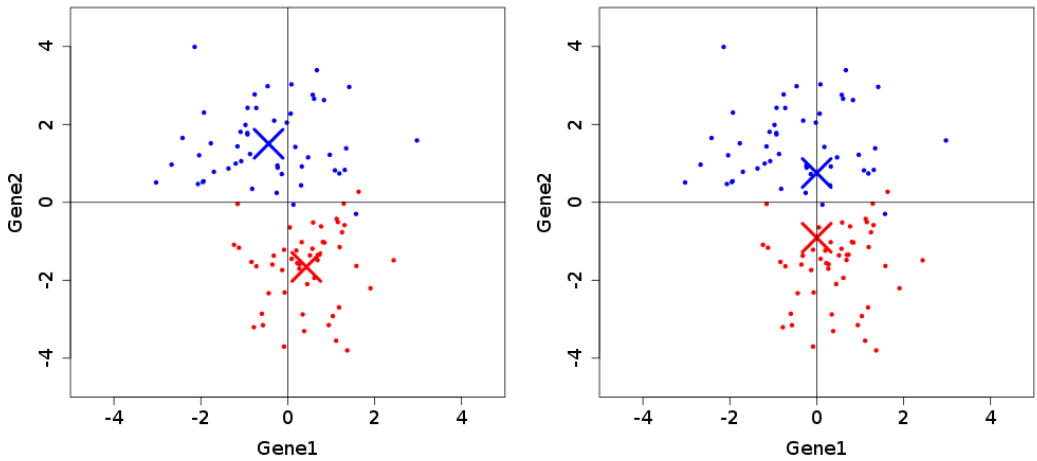
How to select hyperparameters?

- Trial and error
- Grid search



- Global optimization solvers (e.g., interalg)

Nearest shrunken centroids (PAMR)

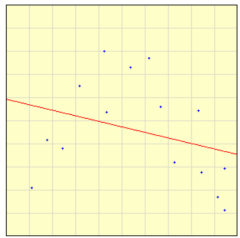
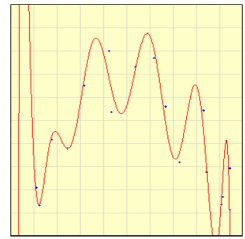
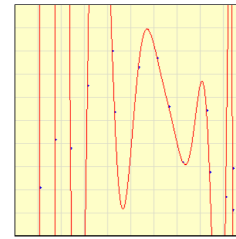


New samples (points) are assigned to the nearest centroid

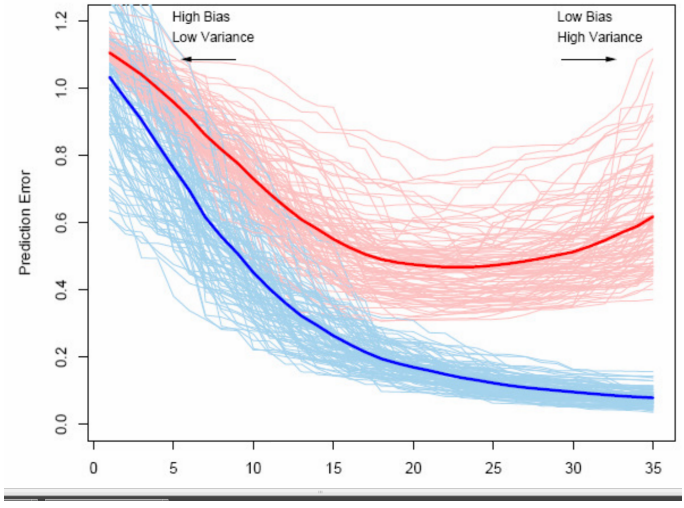
- by sum of squared distances
- genes with all zero centroids are ignored

Bias-variance tradeoff

- More complex classifier => less bias, but more variance

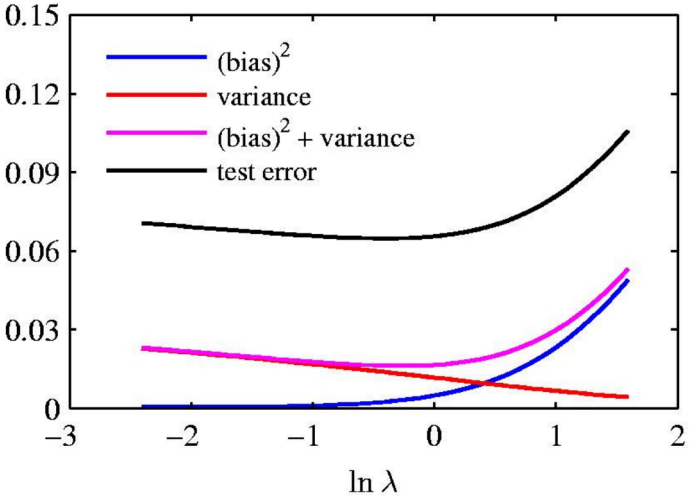
Bias-variance tradeoff



- Experiment repeated 100 times
- Blue = Training Error, Red = Test Error

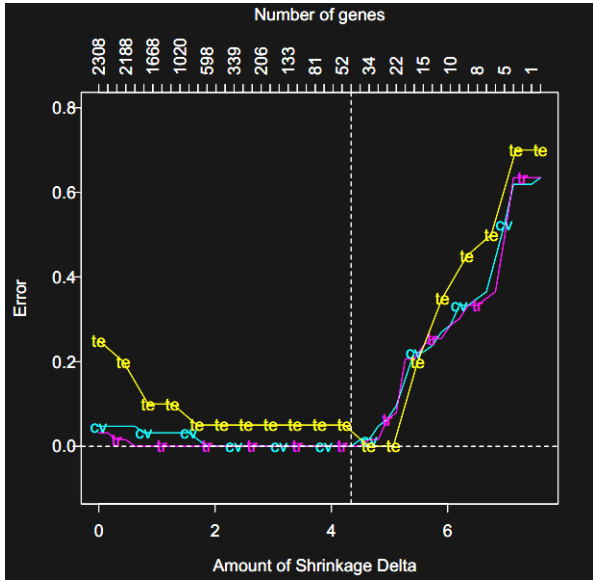
- Message: lower training error is not always better!

Bias-variance tradeoff



Bias-variance tradeoff in SVM and PAMR

- For PAMR, the regularization hyperparameter is the **threshold** (or delta).
- In PAMR, it is autoselected by cross-validation
- In SVM, it is "C"


Number of genes
2308
2188
1668
1020
598
339
206
133
81
52
34
22
15
10
8
5
1
Error
0.0
0.2
0.4
0.6
0.8
0
2
4
6
Amount of Shrinkage Delta
te
cv
tr

Feature selection

- **Feature selection** increases bias of a classifier, and thus improves its generalizability. When people speak of "N-gene" signatures, they have chosen the N (hopefully nonredundant and most informative) genes by feature selection.
- PAMR performs feature selection automatically. The more stringent the threshold, the fewer genes will be selected.
- SVM does NOT perform feature selection. "penalizedSVM" package, however, will do this.

- For linear models, "lasso" does feature selection. R package "lars" (not covered here).

Feature selection

- Typically, the feature selection algo will be specific to the ML algo. However, one simple and popular method is "recursive feature elimination": greedily remove one variable at a time until you stop seeing performance improvement.
- Relatively **poor** way to perform feature selection is by choosing the N predictors most correlated with response. This neglects covariance.

Assessing performance

In 2-class problem:

- Accuracy: (TP + TN) / (FP + FN)
- Precision: TP / (TP + FP)
- Recall: TP / (TP + FN)
- **F-measure: weighted average between Precision and Recall**
    - (usually even weighting)
- **ROC AUC**

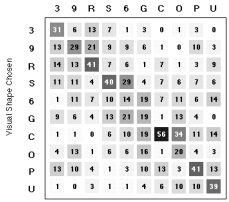
- Plot TPR vs FPR
- Integrate area

- Lift: evaluate fraction improvement over random guessing for top N% predictions

The performance metric you use should reflect the application!

Multi-class performance assessment is considerably more complicated.

Assessing performance: confusion matrices



Data splits

- Training set: train the classifier.
- Development test: Evaluate classifier's
- Test set: Use only at the end of algorithm and parameter selection. Purpose: to evaluate performance.

80%/10%/10% is a common split.

If you are using cross-validation, only need training and test, not development.

Summary of machine learning steps

1. Acquire predictor and response variables (expression and phenotype, usually)
2. Perform feature selection (if algorithm doesn't do feature selection for you)
3. Train classifier with "training data" and particular hyperparameters
4. Evaluate classifier performance on "development test" data
5. Repeat steps 3-4 via grid search to find optimal hyperparameters.

6. Evaluate classifier performance on test set to approximate real-world performance.

General advice for using machine learning

- Understand the algorithm. Be **very** sure to understand the bias-variance tradeoff.
- Use cross-validation. In critical scenarios, use training, validation, and test sets.
- Do frequent sanity checks and manual inspection of the predictions (high-profile "flip-flops" have occurred).
- "Black-box" methods such as SVMs slightly outperform simpler linear, logistic, or nearest neighbor models, but at the expense of model

comprehensibility. Clinicians or users often want to know **why** a sample was classified the way it was.

- Try lots of performance metrics, but pay most attention to the one relevant to *your* application.

Further topics (not covered)

If you are interested in this topic, check out also:

- Random forests
- Boosting
- Ensemble classifiers
- Naive Bayes and Bayes networks