Listening to what genes are saying

Statistical learning from gene expression data

Dennis Wylie, UT Bioinformatics Consulting Group

February 27, 2015

Outline

1 Introduction

2 Normalization

3 Feature selection

4 Classification

5 knn

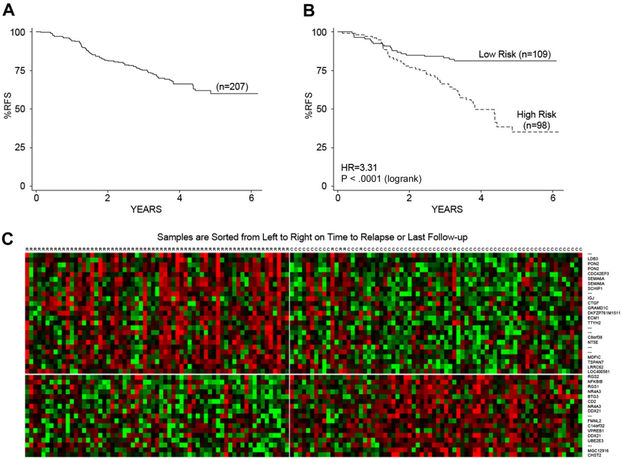
6 Linear models

7 Naive Bayes

8 SVM

9 Other methods

What is a classifier?



A 38-gene expression classifier predictive of relapse-free survival (RFS) could distinguish 2 groups with differing relapse risks: low (4-year RFS, 81%, n = 109) versus high (4-year RFS, 50%, n = 98; P < .001).

Taken from Kang *et al.* (2010).

Classification by gene expression

Goal:
Given sample i, use measured gene expression levels $x_{ig} \in \mathbb{R}$ for $g \in \{1, \ldots, p\}$ to assign class label y_i.

Use vector notation $\mathbf{x}_i$ to represent collection of all gene measurements x_{ig} for sample i.

To keep things simple, consider only two-class problems (say, "low-risk" vs. "high-risk") so that $y_i \in \{0, 1\}$.

Define random variables $\mathbf{X}$ and Y of which $\mathbf{x}_i$ and y_i will be regarded as particular realizations.

Model should yield $\mathbb{P}(Y = y \mid \mathbf{X} = \mathbf{x}) \ldots$

Training and test sets

Select modeling strategy M and apply algorithm to find parameters $\boldsymbol{\theta}$ using a set S_{train} of samples such that

$$\mathbb{P}_{M,\boldsymbol{\theta}}(Y = y_i \mid \mathbf{X} = \mathbf{x}_i)$$

has high probability for the observed class labels y_i for $i \in S_{\text{train}}$.

Training and test sets

Select modeling strategy M and apply algorithm to find parameters $\boldsymbol{\theta}$ using a set S_{train} of samples such that

$$\mathbb{P}_{M,\boldsymbol{\theta}}(Y = y_i \mid \mathbf{X} = \mathbf{x}_i)$$

has high probability for the observed class labels y_i for $i \in S_{\text{train}}$.

However, what we really want is for model to accurately classify samples $j \notin S_{\text{train}}$ whose true classifications y_j may not already be known.

Generally $(M, \boldsymbol{\theta})$ will not perform as well on samples $j \notin S_{\text{train}}$ as it does on $i \in S_{\text{train}}$.

Thus useful to apply $(M, \boldsymbol{\theta})$ to $j \in S_{\text{test}}$ where $S_{\text{test}} \cap S_{\text{train}} = \emptyset$ but where the $\{y_j \mid j \in S_{\text{test}}\}$ are still known.

Overfitting I: Resubstitution

Illustration of **overfitting**:

For $i \in \{1, \ldots, 100\}$, simulated data $\mathbf{x}_i$ on pseudogenes $g \in \{1, \ldots, 2500\}$ from $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu} = \mathbf{0}, \Sigma = I)$.

Generated class labels y_i from $Y \sim \text{Bern}(p = 0.5)$ *independently* of $\mathbf{X}$.

Then selected top $n \in \{10, 25, 50, 100\}$ genes by t-test and fit variety of classification models for Y using these genes. . .

Overfitting I: Resubstitution

Illustration of **overfitting**:

For $i \in \{1, \ldots, 100\}$, simulated data $\mathbf{x}_i$ on pseudogenes $g \in \{1, \ldots, 2500\}$ from $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu} = \mathbf{0}, \Sigma = I)$.

Generated class labels y_i from $Y \sim \text{Bern}(p = 0.5)$ *independently* of $\mathbf{X}$.

Then selected top $n \in \{10, 25, 50, 100\}$ genes by t-test and fit variety of classification models for Y using these genes...

Modeling Strategy	AUC	Accuracy	Sensitivity	Specificity	PPV	NPV
t-Test 10: Knn	0.943	0.82	0.893	0.727	0.806	0.842
t-Test 25: Knn	0.960	0.89	0.964	0.795	0.857	0.946
t-Test 50: Knn	0.985	0.92	0.982	0.841	0.887	0.974
t-Test 100: Knn	0.999	0.97	1.000	0.932	0.949	1.000
t-Test 10: Logistic	0.933	0.88	0.911	0.841	0.879	0.881
t-Test 25: Logistic	0.996	0.97	1.000	0.932	0.949	1.000
t-Test 50: Logistic	1.000	1.00	1.000	1.000	1.000	1.000
t-Test 100: Logistic	1.000	1.00	1.000	1.000	1.000	1.000
t-Test 10: Svm	0.981	0.92	0.946	0.886	0.914	0.929
t-Test 25: Svm	1.000	0.99	1.000	0.977	0.982	1.000
t-Test 50: Svm	1.000	1.00	1.000	1.000	1.000	1.000
t-Test 100: Svm	1.000	1.00	1.000	1.000	1.000	1.000

Metrics—Binomial

There are many ways to measure performance for classifiers; most are based only on the "discretized calls" $\hat{y}$

$$\hat{y}_{M,\boldsymbol{\theta},\psi} = \begin{cases} 1 & \text{if } \mathbb{P}_{M,\boldsymbol{\theta}}(Y = 1 \mid \mathbf{X} = \mathbf{x}) \geq \psi \\ 0 & \text{otherwise} \end{cases}$$

given some threshold ψ (e.g., $\psi = 0.5$).

Metrics—Binomial

There are many ways to measure performance for classifiers; most are based only on the "discretized calls" $\hat{y}$

$$\hat{y}_{M,\boldsymbol{\theta},\psi} = \begin{cases} 1 & \text{if } \mathbb{P}_{M,\boldsymbol{\theta}}(Y = 1 \mid \mathbf{X} = \mathbf{x}) \geq \psi \\ 0 & \text{otherwise} \end{cases}$$

given some threshold ψ (e.g., $\psi = 0.5$).
Given a sample set S of size $|S| = N$ composed of:

TP true positive samples: $y = \hat{y} = 1$

TN true negative samples: $y = \hat{y} = 0$

FP false positive samples: $y = 0, \hat{y} = 1$

FN false negative samples: $y = 1, \hat{y} = 0$,

define

Accuracy fraction of calls correct $\left(\frac{TP+TN}{N}\right)$

Sensitivity fraction of calls correct when $y = 1$ $\left(\frac{TP}{TP+FN}\right)$

Specificity fraction of calls correct when $y = 0$ $\left(\frac{TN}{TN+FP}\right)$

PPV fraction of calls correct when $\hat{y} = 1$ $\left(\frac{TP}{TP+FP}\right)$

NPV fraction of calls correct when $\hat{y} = 0$ $\left(\frac{TN}{TN+FN}\right)$.

Metrics—Receiver Operating Characteristic (ROC)

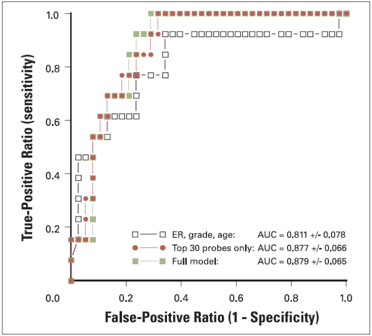


Fig 3. Receiver operating characteristic curves of three distinct pathologic complete response prediction models. The performance of the Diagonal Linear Discriminant Analysis–30 predictor and a predictor based on clinical variables and a combined clinical + pharmacogenomic prediction model are shown in the validation set (n = 51). ER, estrogen receptor; AUC, area under the curve.

Taken from Hess *et al.* (2006).

Could consider binomial metrics over range of threshold values ψ.

Receiver operating characteristic (ROC) curve does this for sensitivity and specificity.

Area under ROC curve (**AUC**) = probability that score $\mathbb{P}(Y = 1 \mid \mathbf{X} = \mathbf{x})$ of a randomly chosen positive case ($y = 1$) is higher than score of a randomly chosen negative case($y = 0$).

Cross-validation (CV)

But what if we don't have a test set S_{test} lying around?

Can always split whatever sample set you have up into a test and training set.

If not many samples available, might split samples S into S_1 and S_2 and then try:

1. first train M on S_1 to obtain parametrized model $(M, \boldsymbol{\theta}_1)$ for testing on S_2;
2. then train on S_2 to obtain model $(M, \boldsymbol{\theta}_2)$ for testing on S_1.

Unbiased performance estimate could then be obtained using the predictions $\mathbb{P}_{M,\boldsymbol{\theta}_2}(Y \mid \mathbf{X})$ for samples in S_1 and predictions $\mathbb{P}_{M,\boldsymbol{\theta}_1}(Y \mid \mathbf{X})$ for samples in S_2.

k-fold cross-validation

This procedure can be generalized to split S up into k subsets S_k for each of which:

1. a model $(M, \boldsymbol{\theta}_{\text{-}k})$ is trained using training set $S_{\text{-}k} = \bigcup_{q \neq k} S_q$
2. predictions $\mathbb{P}_{M,\boldsymbol{\theta}_{\text{-}k}}(Y \mid \mathbf{X} = \mathsf{x}_i)$ are made for samples $i \in S_k$.

k-fold cross-validation

This procedure can be generalized to split S up into k subsets S_k for each of which:

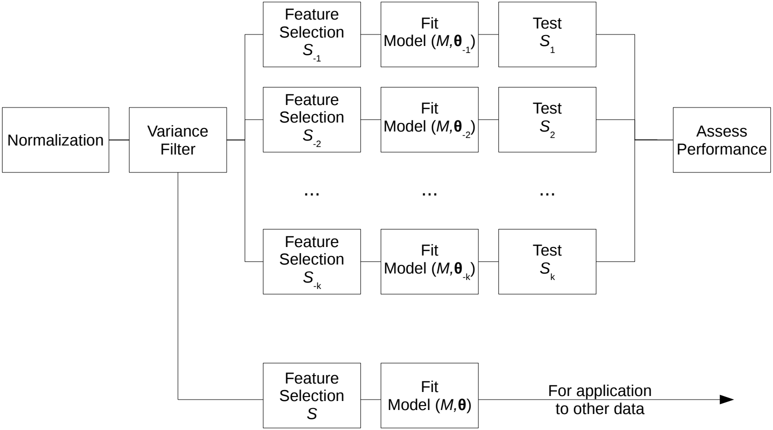
1. a model $(M, \boldsymbol{\theta}_{\text{-}k})$ is trained using training set $S_{\text{-}k} = \bigcup_{q \neq k} S_q$
2. predictions $\mathbb{P}_{M,\boldsymbol{\theta}_{\text{-}k}}(Y \mid \mathbf{X} = \mathsf{x}_i)$ are made for samples $i \in S_k$.

Very important:
cross-validation is only valid if all *supervised* steps performed in building a classification model are conducted separately in each of the k-folds.

I'm looking at you, feature selection!

The `crossval` function in the `R` package `bootstrap` can do k-fold cross-validation if you wrap the entire modeling procedure in an `R` function.

k-fold cross-validation


Normalization
Variance Filter
Feature Selection S_{-1}
Fit Model ($M,\boldsymbol{\theta}_{-1}$)
Test S_1
Feature Selection S_{-2}
Fit Model ($M,\boldsymbol{\theta}_{-2}$)
Test S_2
...
...
...
Feature Selection S_{-k}
Fit Model ($M,\boldsymbol{\theta}_{-k}$)
Test S_k
Assess Performance
Feature Selection S
Fit Model ($M,\boldsymbol{\theta}$)
For application to other data

- Depending on details of modeling strategy M, unsupervised normalization and variance filtration may be done outside CV
- CV assesses performance of modeling strategy M, *not* of the specific parametrized model $(M, \boldsymbol{\theta})$.

Overfitting II: Cross-validation

Returning to overfit example. . .

For $i \in \{1, \ldots, 100\}$, simulated data $\mathbf{x}_i$ on pseudogenes $g \in \{1, \ldots, 2500\}$ from $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu} = \mathbf{0}, \Sigma = I)$.

Generated class labels y_i from $Y \sim \text{Bern}(p = 0.5)$ *independently* of $\mathbf{X}$.

Then selected top $n \in \{10, 25, 50, 100\}$ genes by t-test and fit variety of classification models for Y using these genes:

Modeling Strategy	AUC	Accuracy	Sensitivity	Specificity	PPV	NPV
t-Test 10: Knn	0.476	0.47	0.589	0.318	0.524	0.378
t-Test 25: Knn	0.558	0.55	0.714	0.341	0.580	0.484
t-Test 50: Knn	0.286	0.41	0.571	0.205	0.478	0.273
t-Test 100: Knn	0.446	0.54	0.768	0.250	0.566	0.458
t-Test 10: Logistic	0.357	0.39	0.464	0.295	0.456	0.302
t-Test 25: Logistic	0.554	0.54	0.643	0.409	0.581	0.474
t-Test 50: Logistic	0.362	0.46	0.571	0.318	0.516	0.368
t-Test 100: Logistic	0.418	0.53	0.768	0.227	0.558	0.435
t-Test 10: Svm	0.376	0.42	0.518	0.295	0.483	0.325
t-Test 25: Svm	0.484	0.50	0.607	0.364	0.548	0.421
t-Test 50: Svm	0.347	0.40	0.482	0.295	0.466	0.310
t-Test 100: Svm	0.464	0.51	0.589	0.409	0.559	0.439

Normalization

Basic measurement unit of RNA-seq is count of reads mapped to a given marker (gene, exon, etc.).

Besides biological expression levels, many technical factors influence these counts as well, e.g.:

1. differences in library size (sequencing depth)
2. length of gene

Normalization

Basic measurement unit of RNA-seq is count of reads mapped to a given marker (gene, exon, etc.).

Besides biological expression levels, many technical factors influence these counts as well, e.g.:

1. differences in library size (sequencing depth)
2. length of gene

Simplest normalization schemes account for these influences by

1. dividing the total library size (and multiplying by 10^6) to obtain CPM or
2. further dividing by gene length (and multiplying by 10^3) to obtain RPKM

(Normalization for gene length may not be necessary in studies which do not attempt to compare expression levels between different genes.)

Alternative normalization methods

Some studies have found that RPKM normalization may not appropriately control for association between gene length and read counts (Dillies *et al.* (2013)).

Further, both CPM and RPKM may overweight influence of few very highly expressed genes which may actually be differentially expressed across samples.

Alternative normalization methods

Some studies have found that RPKM normalization may not appropriately control for association between gene length and read counts (Dillies *et al.* (2013)).

Further, both CPM and RPKM may overweight influence of few very highly expressed genes which may actually be differentially expressed across samples.

Simple alternatives are to use upper quartile- or median-read count as sample normalization factor instead of sum; Dillies *et al.* (2013) found these options preferable.

Alternative normalization methods

Some studies have found that RPKM normalization may not appropriately control for association between gene length and read counts (Dillies *et al.* (2013)).

Further, both CPM and RPKM may overweight influence of few very highly expressed genes which may actually be differentially expressed across samples.

Simple alternatives are to use upper quartile- or median-read count as sample normalization factor instead of sum; Dillies *et al.* (2013) found these options preferable.

More complex normalization methods offered by the `R` packages `DESeq` and `edgeR`; may offer better performance in some circumstances.

Feature selection

Generally assumed that expression patterns of most genes are either:

1. uninformative or
2. contain only information redundant with a small number of maximally useful markers

with respect to a particular classification task.

Feature selection attempts to identify optimal set of markers for inclusion in classifier.

Not all modeling techniques absolutely require upfront feature selection but the resulting simplification:

1. reduces computational workload,
2. can help to avoid overfitting (though feature selection can itself be susceptible to overfitting), and
3. facilitates model platform migration.

Taxonomy (adapted from Saeys *et al.* (2007))

Filter Selection done before and independently of classifier construction. Can be univariate or multivariate.

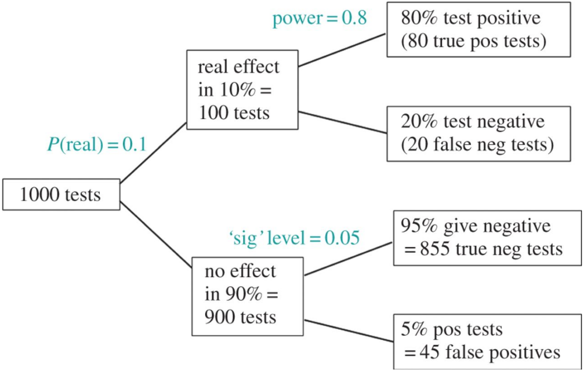
Wrapper Embed classifier construction within feature selection process. Heuristic search methods compare models, favor adding or removing features based on optimization of some specified metric on resulting classifiers.

Embedded Feature selection is inherently built into some classifier construction methods.

Taxonomy (adapted from Saeys *et al.* (2007))

Category	Advantages	Disadvantages	Examples
Filter	*Univariate*		
	Fast Scalable Independent of classifier	- feature dependencies - interaction w/classifier	t-test, ANOVA Wilcox test Rank Product
	Multivariate		
	+ feature dependencies Independent of classifier Intermediate complexity	Slower Less Scalable - interaction w/classifier	CFS Markov Blanket Filter
Wrapper	*Deterministic*		
	Simple + interaction w/classifier + feature dependencies	Risk of over-fitting Greedy (local optima) Classifier dependent selection	Forward Selection Backward Elimination Plus q minus r
	Randomized		
	Less prone to local optima + interaction w/classifier + feature dependencies	High risk over-fitting Computationally intensive Classifier dependent selection	Simulated Annealing Randomized Hill Climbing Genetic Algorithms
Embedded	+ interaction w/classifier + feature dependencies Intermediate complexity	No modularity Restrict algorithms	Decision trees Weighted Naive Bayes LASSO regression

False Discovery Rate (FDR)



1000 tests

$P(\text{real}) = 0.1$

real effect in 10% = 100 tests

power = 0.8

80% test positive (80 true pos tests)

20% test negative (20 false neg tests)

no effect in 90% = 900 tests

'sig' level = 0.05

95% give negative = 855 true neg tests

5% pos tests = 45 false positives

Tree diagram to illustrate the false discovery rate in significance tests. This example considers 1000 tests, in which the prevalence of real effects is 10%. The lower limb shows that with the conventional significance level, p=0.05, there will be 45 false positives. The upper limb shows that there will be 80 true positive tests. The false discovery rate is therefore 45/(45+80)=36%, far bigger than 5%.

Taken from Colquhoun (2014).

False Discovery Rate (FDR)

When statistical hypothesis testing employed for feature selection, multiple comparisons problem must be confronted:

With p markers, even if very few truly differentially expressed, $\approx \alpha p$ false positive results will be obtained.

False Discovery Rate (FDR)

When statistical hypothesis testing employed for feature selection, multiple comparisons problem must be confronted:

With p markers, even if very few truly differentially expressed, $\approx \alpha p$ false positive results will be obtained.

Many methods proposed to mitigate; most popular is **false discovery rate (FDR)** method of Benjamini & Hochberg (1995) (implemented in `R` by function `p.adjust` with argument `method="fdr"`).

Idea: control the fraction of reported positive (significant) results which are really false positives.

False Discovery Rate (FDR)

When statistical hypothesis testing employed for feature selection, multiple comparisons problem must be confronted:

With *p* markers, even if very few truly differentially expressed, $\approx \alpha p$ false positive results will be obtained.

Many methods proposed to mitigate; most popular is **false discovery rate (FDR)** method of Benjamini & Hochberg (1995) (implemented in `R` by function `p.adjust` with argument `method="fdr"`).

Idea: control the fraction of reported positive (significant) results which are really false positives.

Multiple comparisons should be taken into account when powering a study as well. E.g., if a particular FDR is targeted, should estimate what unadjusted single test significance level α might yield that FDR.

Variance filtration

Markers with weak effect sizes but apparently low within-group variance tend to make up large number of false positives.

Variance filtration

Markers with weak effect sizes but apparently low within-group variance tend to make up large number of false positives.

Small between-group effect + low within-group variance $\Longrightarrow$ low overall variance. Overall variance $\mathbb{V}[X_g]$ of a marker is independent of the class Y.

For many statistical hypothesis tests, such independent filtering steps can be performed prior to testing to reduce comparisons which must be accounted for by FDR (Bourgon *et al.* (2010)).

Variance filtration

Markers with weak effect sizes but apparently low within-group variance tend to make up large number of false positives.

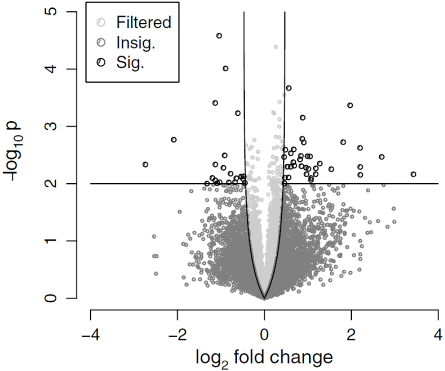
Small between-group effect + low within-group variance $\implies$ low overall variance. Overall variance $\mathbb{V}[X_g]$ of a marker is independent of the class Y.

For many statistical hypothesis tests, such independent filtering steps can be performed prior to testing to reduce comparisons which must be accounted for by FDR (Bourgon *et al.* (2010)).

Such Y-independent filtering steps are only forms of feature selection that may be applied outside cross-validation (Hastie *et al.* (2009))

. . . though should always take care that "independent filtering" step really is independent, as discussed in Bourgon *et al.* (2010).

Variance filtration



Overall variance (or equivalently, overall standard deviation) filtering example, using the ALL data, comparing 3 BCR/ABL and 3 control subjects. (A) Volcano plot contrasting log-fold change with *p*-value, as obtained from a standard *t*-test. The impact of filtering is shown: overall variance filtering is equivalent to requiring a minimum fold change—where the bound increases as the *p*-value decreases.

Taken from Bourgon *et al.* (2010).

Empirical Bayes

Multiple comparisons aren't all bad news for statistical inference.

When many similar variables simultaneously measured, possible to "borrow information" across variables.

Empirical Bayes

Multiple comparisons aren't all bad news for statistical inference.

When many similar variables simultaneously measured, possible to "borrow information" across variables.

Empirical Bayes methods (Efron (2010)) mix frequentist and Bayesian ideas to empirically estimate something like a Bayesian prior for distributional parameters of individual genes.

Can be derived as approximations to fully Bayesian hierarchical models.

Empirical Bayes

Multiple comparisons aren't all bad news for statistical inference.

When many similar variables simultaneously measured, possible to "borrow information" across variables.

Empirical Bayes methods (Efron (2010)) mix frequentist and Bayesian ideas to empirically estimate something like a Bayesian prior for distributional parameters of individual genes.

Can be derived as approximations to fully Bayesian hierarchical models.

The `R` package `limma` (Ritchie *et al.* (2015)) uses these ideas to identify differentially expressed genes with fewer false positives;

achieved largely through shrinking individual gene variance estimates towards a pooled variance estimate (so may not be compatible with variance filtration).

Empirical Bayesball

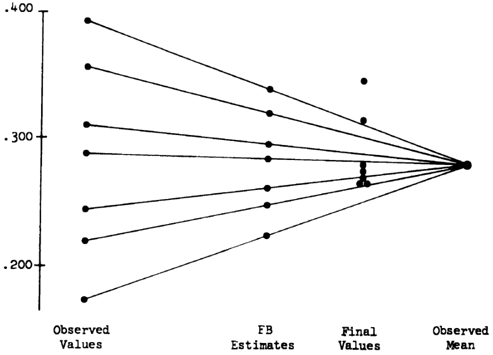


Figure 3. Graphical Display of the Baseball Data.

Observed Values batting averages from first 45 at-bats

EB Estimates shrink Observed Values towards mean

Final values final batting average for player

Observed Mean use your imagination.

Taken from Casella (1985).

Supervised learning

Feature selection often identifies markers g for which the class labels Y predict X_g through $\mathbb{P}(X_g \mid Y)$.

Supervised learning

Feature selection often identifies markers g for which the class labels Y predict X_g through $\mathbb{P}(X_g \mid Y)$.

Classification seeks to use feature data $\mathbf{X} = \mathbf{x}$ to predict Y through $\mathbb{P}(Y \mid \mathbf{X})$.

Supervised learning

Feature selection often identifies markers g for which the class labels Y predict X_g through $\mathbb{P}(X_g \mid Y)$.

Classification seeks to use feature data $\mathbf{X} = \mathbf{x}$ to predict Y through $\mathbb{P}(Y \mid \mathbf{X})$.

Supervised classification uses a *training set* $\{(\mathbf{x}_i, y_i) \mid i \in \{1, ..., N\}\}$ to construct a classifier $M, \boldsymbol{\theta}$ which can be used to make predictions $P_{M,\boldsymbol{\theta}}(Y = y \mid \mathbf{X} = \mathbf{x})$.

k-nearest-neighbors (knn)

Perhaps simplest approach to classification:

***k*-nearest neighbors**
Given vector $\mathbf{x}$ of feature values (e.g., expression counts x_g for selected genes g) with k nearest training vectors

$$\{\mathbf{x}_j \mid j \in \mathrm{NN}_k\},$$

with $\|\mathbf{x}_j - \mathbf{x}\| \leq \|\mathbf{x}_i - \mathbf{x}\|$ if $j \in \mathrm{NN}_k$ and $i \notin \mathrm{NN}_k$:

$$\mathbb{P}(Y = 1 \mid X = x) = \frac{1}{|\mathrm{NN}_k|} \sum_{j \in \mathrm{NN}_k} y_j$$

k-nearest-neighbors (knn)

Perhaps simplest approach to classification:

***k*-nearest neighbors**
Given vector $\mathbf{x}$ of feature values (e.g., expression counts x_g for selected genes g) with k nearest training vectors

$$\{\mathbf{x}_j \mid j \in \mathrm{NN}_k\},$$

with $\|\mathbf{x}_j - \mathbf{x}\| \leq \|\mathbf{x}_i - \mathbf{x}\|$ if $j \in \mathrm{NN}_k$ and $i \notin \mathrm{NN}_k$:

$$\mathbb{P}(Y = 1 \mid X = x) = \frac{1}{|\mathrm{NN}_k|} \sum_{j \in \mathrm{NN}_k} y_j$$

As long as there is natural metric on feature space, this method has a lot to recommend it in low-dimensional settings.

k-nearest-neighbors is implemented in `R` by the `knn` function from the package `class`.

knn and the curse of dimensionality

Volume of p-dimensional hypersphere of radius r is

$$V_p(r) = \frac{\pi^{p/2}}{\Gamma\left(\frac{p}{2}+1\right)} r^p \propto r^p$$

For $\mathbf{x}$ to have many neighbors nearer than r, must be many $\mathbf{x}_i \in S_{\text{train}}$ in volume $V_p(r)$ centered at $\mathbf{x}$.

knn and the curse of dimensionality

Volume of p-dimensional hypersphere of radius r is

$$V_p(r) = \frac{\pi^{p/2}}{\Gamma\left(\frac{p}{2}+1\right)} r^p \propto r^p$$

For $\mathbf{x}$ to have many neighbors nearer than r, must be many $\mathbf{x}_i \in S_{\text{train}}$ in volume $V_p(r)$ centered at $\mathbf{x}$.

If the dimensionality p is large and r is small, this is very unlikely.

So must use points far away to guess what's going on at $\mathbf{x}$.

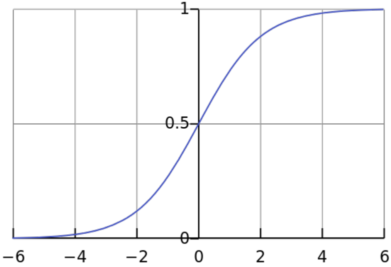
Not surprisingly this doesn't always work.

Linear models

In the context of classification, "linear model" usually means

$$\mathbb{P}(Y = 1 \mid \mathbf{X} = \mathbf{x}) = \text{expit}(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x})$$

where expit: $\mathbb{R} \to (0,1)$ defined by $\text{expit}(u) = \frac{\exp(u)}{1+\exp(u)}$ is the logistic, or inverse-logit, function.



Linear models

In the context of classification, “linear model” usually means

$$\mathbb{P}(Y = 1 \mid \mathbf{X} = \mathbf{x}) = \text{expit}(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x})$$

where expit: $\mathbb{R} \to (0, 1)$ defined by $\text{expit}(u) = \frac{\exp(u)}{1+\exp(u)}$ is the logistic, or inverse-logit, function.

Two main classes of such linear classification models:

1. **linear discriminant analysis (LDA)** (marginal): adds assumption $\mathbb{P}(\mathbf{X} = \mathbf{x} | Y = y) \sim \mathcal{N}(\mu_y, \Sigma)$
2. **logistic regression** (conditional): makes no explicit distributional assumptions about $\mathbf{X}$, instead maximizes likelihood of conditional $\mathbb{P}(Y \mid \mathbf{X})$ over training set.

Linear models in high-dimensional settings

While less flexible than knn, linear models can be made robust in high-dimensional settings using **regularization**.

Unregularized linear regression uses maximum likelihood to select coefficients β_g; fit by ordinary least-squares (OLS) estimator:

$$\hat{\boldsymbol{\beta}}_{\mathsf{OLS}} = \arg\min_{\boldsymbol{\beta}} \sum_i (y_i - \boldsymbol{\beta} \cdot \mathsf{x}_i)^2$$

Bayesian derivation of OLS uses uniform prior on $\boldsymbol{\beta}$.

Linear models in high-dimensional settings

While less flexible than knn, linear models can be made robust in high-dimensional settings using **regularization**.

Unregularized linear regression uses maximum likelihood to select coefficients β_g; fit by ordinary least-squares (OLS) estimator:

$$\hat{\boldsymbol{\beta}}_{\mathrm{OLS}} = \arg\min_{\boldsymbol{\beta}} \sum_i (y_i - \boldsymbol{\beta} \cdot \mathrm{x}_i)^2$$

Bayesian derivation of OLS uses uniform prior on $\boldsymbol{\beta}$.

If instead Gaussian prior (**L2 regression**) imposed on $\boldsymbol{\beta}$, maximum a posteriori (MAP) estimator is (Park & Casella (2008)):

$$\hat{\boldsymbol{\beta}}_{\mathrm{L2}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_i (y_i - \boldsymbol{\beta} \cdot \mathrm{x}_i)^2 + \phi_2 \sum_g \beta_g^2 \right]$$

where the regularization parameter ϕ_2 determined by variance of the Gaussian prior.

Lasso regression

Alternatively, use of Laplace prior for $\boldsymbol{\beta}$ yields MAP estimator (Park & Casella (2008)):

$$\hat{\boldsymbol{\beta}}_{\mathrm{L1}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_i (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \phi_1 \sum_g |\beta_g| \right]$$

where now ϕ_1 is determined by width of the Laplace prior.

Lasso regression

Alternatively, use of Laplace prior for $\boldsymbol{\beta}$ yields MAP estimator (Park & Casella (2008)):

$$\hat{\boldsymbol{\beta}}_{\mathsf{L1}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_i (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \phi_1 \sum_g |\beta_g| \right]$$

where now ϕ_1 is determined by width of the Laplace prior.

As ϕ_1 is increased, progressively more β_g set to zero, de-selecting the corresponding features (Tibshirani (1996))— **L1, or LASSO, regression** is an embedded feature selection method.

Lasso regression

Alternatively, use of Laplace prior for $\boldsymbol{\beta}$ yields MAP estimator (Park & Casella (2008)):

$$\hat{\boldsymbol{\beta}}_{\mathsf{L1}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_i (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \phi_1 \sum_g |\beta_g| \right]$$

where now ϕ_1 is determined by width of the Laplace prior.

As ϕ_1 is increased, progressively more β_g set to zero, de-selecting the corresponding features (Tibshirani (1996))— **L1, or LASSO, regression** is an embedded feature selection method.

Both L1/LASSO and L2/ridge logistic regression are implemented in the `R` package `glmnet` function `glmnet` using argument `family="binomial"`.

Shrinkage and diagonal LDA

LDA can also be regularized:
Instead of using the maximum likelihood estimator for the covariance matrix Σ, off-diagonal entries Σ_{gh} are shrunk by a regularization parameter towards 0 (`R` package `sda`).

Shrinkage and diagonal LDA

LDA can also be regularized:
Instead of using the maximum likelihood estimator for the covariance matrix Σ, off-diagonal entries Σ_{gh} are shrunk by a regularization parameter towards 0 (`R` package `sda`).

In most extreme form, shrinkage LDA sets all $\Sigma_{gh} = 0$ ($g \neq h$).

Since Σ is now a diagonal matrix, this is referred to as diagonal LDA or **DLDA**. DLDA has been been found to be particularly useful for gene expression data (Dudoit *et al.* (2002)).

A nice implementation of DLDA can be found in the `dlda` function in the `R` package `sparsediscrim`.

Naive Bayes

"Naive Bayes" describes a family of classification methods sharing a common assumption:

$$\mathbb{P}(\mathbf{X} = \mathsf{x} \mid Y = y) = \prod_{g} \mathbb{P}(X_g = x_g \mid Y = y)$$

which can be substituted into Bayes' formula to yield:

$$\mathbb{P}(Y = y \mid \mathbf{X} = \mathsf{x}) = \frac{\prod_{g} \mathbb{P}(X_g = x_g \mid Y = y)}{\sum_{y'} \prod_{g} \mathbb{P}(X_g = x_g \mid Y = y')}$$

Naive Bayes

“Naive Bayes” describes a family of classification methods sharing a common assumption:

$$\mathbb{P}(\mathbf{X} = \mathsf{x} \mid Y = y) = \prod_g \mathbb{P}(X_g = x_g \mid Y = y)$$

which can be substituted into Bayes’ formula to yield:

$$\mathbb{P}(Y = y \mid \mathbf{X} = \mathsf{x}) = \frac{\prod_g \mathbb{P}(X_g = x_g \mid Y = y)}{\sum_{y'} \prod_g \mathbb{P}(X_g = x_g \mid Y = y')}$$

DLDA is actually a form of naive Bayes classification in which the additional assumption of linearity is posed.

Naive Bayes: does it work?

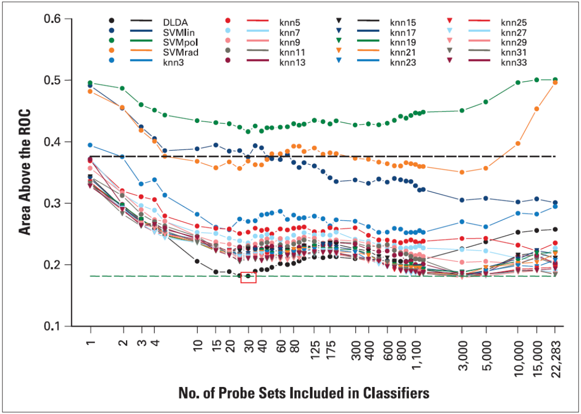


Fig 1. Mean area above the receiver operating characteristic (ROC) curves plotted against the number of top genes included in the classifiers. Complete 5-fold cross validation results (means over the 100 iterations) for 20 classifier algorithms including different numbers of probe sets (39 gene sets) are shown. Green and black horizontal dotted lines indicate the mean +/− 2SD for the nominally best Diagonal Linear Discriminant Analysis (DLDA) classifier with 30 probe sets that was selected for independent validation. polynomial kernels (SVM), and K-nearest neighbor

Taken from Hess *et al.* (2006).

Naive Bayes: does it work?

The conditional independence assumption is basically never true, but:

1. frequently not enough data to accurately assess true inter-feature covariance, so that attempts to do so just lead to overfitting, and

Naive Bayes: does it work?

The conditional independence assumption is basically never true, but:

1. frequently not enough data to accurately assess true inter-feature covariance, so that attempts to do so just lead to overfitting, and
2. while this assumption tends to lead to **overconfident** classifiers—probability scores very near 0 or 1 even when wrong—it still often leads to **accurate** classifiers—most calls aren't wrong.

Naive Bayes: does it work?

The conditional independence assumption is basically never true, but:

1. frequently not enough data to accurately assess true inter-feature covariance, so that attempts to do so just lead to overfitting, and
2. while this assumption tends to lead to **overconfident** classifiers—probability scores very near 0 or 1 even when wrong—it still often leads to **accurate** classifiers—most calls aren't wrong.
3. Naive Bayes methods work well when either:
   - features truly are independent within each class *or*
   - features are very tightly correlated (may actually be more relevant in gene expression context) (Rish *et al.* (2001)).

Bias-variance tradeoff

From Wikipedia (http://en.wikipedia.org/wiki/Bias–variance_tradeoff):

> The bias–variance tradeoff (or dilemma) is the problem of simultaneously minimizing two sources of error that prevent supervised learning algorithms from generalizing beyond their training set:
>
> **bias** error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (**underfitting**).
>
> **variance** error from sensitivity to small fluctuations in the training set. High variance can cause **overfitting**: modeling the random noise in the training data, rather than the intended outputs.

Support vector machines (SVMs)

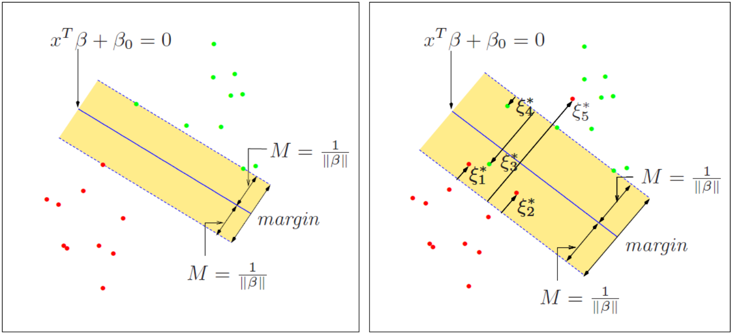


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

Taken from Hastie *et al.* (2009).

Nonlinear SVMs

Can fit SVM in nonlinearly transformed feature space.

For certain transformations, so-called "kernel trick" can be used to do this in very computationally efficient manner. Given a particular transformation h, the kernel

$$k(\mathsf{x}, \mathsf{x}') = \langle h(\mathsf{x}), h(\mathsf{x}') \rangle$$

is actually all that is needed to fit SVM.

Nonlinear SVMs

Can fit SVM in nonlinearly transformed feature space.

For certain transformations, so-called "kernel trick" can be used to do this in very computationally efficient manner. Given a particular transformation h, the kernel

$$k(\mathsf{x}, \mathsf{x}') = \langle h(\mathsf{x}), h(\mathsf{x}') \rangle$$

is actually all that is needed to fit SVM.

Most popular h is rather involved transformation designed to produce the radial basis kernel

$$k(\mathsf{x}, \mathsf{x}') = \exp\left(-\gamma \|\mathsf{x} - \mathsf{x}'\|^2\right)$$

SVMs may be intuitively thought of as classifying a sample with features x based on the (known) classes of similar training data x_i, where "similarity" is quantified by the kernel $k(\mathsf{x}, \mathsf{x}_i)$.

Other methods

- Quadratic discriminant analysis (QDA)
- Decision trees
  - Random forests
  - Boosted trees
- Neural networks
- Graphical models
  - Undirected graphical models
  - Directed acyclic graphs (Bayesian networks)

References I

Benjamini, Yoav, & Hochberg, Yosef. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 289–300.

Bourgon, Richard, Gentleman, Robert, & Huber, Wolfgang. 2010. Independent filtering increases detection power for high-throughput experiments. *Proceedings of the National Academy of Sciences*, **107**(21), 9546–9551.

Casella, George. 1985. An introduction to empirical Bayes data analysis. *The American Statistician*, **39**(2), 83–87.

Colquhoun, David. 2014. An investigation of the false discovery rate and the misinterpretation of p-values. *Royal Society Open Science*, **1**(3), 140216.

Dillies, Marie-Agnès, Rau, Andrea, Aubert, Julie, Hennequet-Antier, Christelle, Jeanmougin, Marine, Servant, Nicolas, Keime, Céline, Marot, Guillemette, Castel, David, Estelle, Jordi, *et al.* 2013. A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. *Briefings in Bioinformatics*, **14**(6), 671–683.

Dudoit, Sandrine, Fridlyand, Jane, & Speed, Terence P. 2002. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, **97**(457), 77–87.

Efron, Bradley. 2010. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction.* Vol. 1. Cambridge University Press.

Hastie, Trevor, Tibshirani, Robert, & Friedman, Jerome. 2009. *The Elements of Statistical Learning.* Springer.

Hess, Kenneth R, Anderson, Keith, Symmans, W Fraser, Valero, Vicente, Ibrahim, Nuhad, Mejia, Jaime A, Booser, Daniel, Theriault, Richard L, Buzdar, Aman U, Dempsey, Peter J, *et al.* 2006. Pharmacogenomic predictor of sensitivity to preoperative chemotherapy with paclitaxel and fluorouracil, doxorubicin, and cyclophosphamide in breast cancer. *Journal of Clinical Oncology*, **24**(26), 4236–4244.

References II

Kang, Huining, Chen, I-Ming, Wilson, Carla S, Bedrick, Edward J, Harvey, Richard C, Atlas, Susan R, Devidas, Meenakshi, Mullighan, Charles G, Wang, Xuefei, Murphy, Maurice, *et al.* 2010. Gene expression classifiers for relapse-free survival and minimal residual disease improve risk classification and outcome prediction in pediatric B-precursor acute lymphoblastic leukemia. *Blood*, **115**(7), 1394–1405.

Park, T., & Casella, G. 2008. The Bayesian lasso. *Journal of the American Statistical Association*, **103**(482), 681–686.

Rish, Irina, Hellerstein, Joseph, & Thathachar, Jayram. 2001. An analysis of data characteristics that affect naive Bayes performance. *IBM TJ Watson Research Center*, **30**.

Ritchie, Matthew E, Phipson, Belinda, Wu, Di, Hu, Yifang, Law, Charity W, Shi, Wei, & Smyth, Gordon K. 2015. limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*, gkv007.

Saeys, Yvan, Inza, Iñaki, & Larrañaga, Pedro. 2007. A review of feature selection techniques in bioinformatics. *Bioinformatics*, **23**(19), 2507–2517.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.