Machine Learning Methods for Gene Expression Data

Day 2

Classification

k-Nearest Neighbors

Overfitting

Cross-Validation

Performance Metrics

Feature Selection

References

Machine Learning Methods for Gene Expression Data

Day 2

Dennis Wylie, UT Bioinformatics Consulting Group

May 21, 2016

Outline

1 Classification

2 *k*-Nearest Neighbors

3 Overfitting

4 Cross-Validation

5 Performance Metrics

6 Feature Selection

Machine Learning Methods for Gene Expression Data

Day 2

Classification

k-Nearest Neighbors

Overfitting

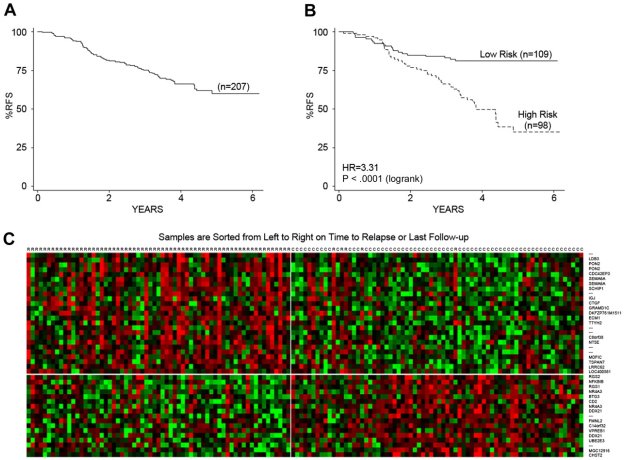
Cross-Validation

Performance Metrics

Feature Selection

References

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 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 x_i and y_i will be regarded as particular realizations.

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

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.

Classification

k-Nearest Neighbors

Overfitting

Cross-Validation

Performance Metrics

Feature Selection

References

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 as:

R	`class::knn`
Python	`sklearn.neighbors.KNeighborsClassifier`

k-nearest-neighbors (knn)

Machine Learning Methods for Gene Expression Data

Day 2

Classification

k-Nearest Neighbors

Overfitting

Cross-Validation

Performance Metrics

Feature Selection

References

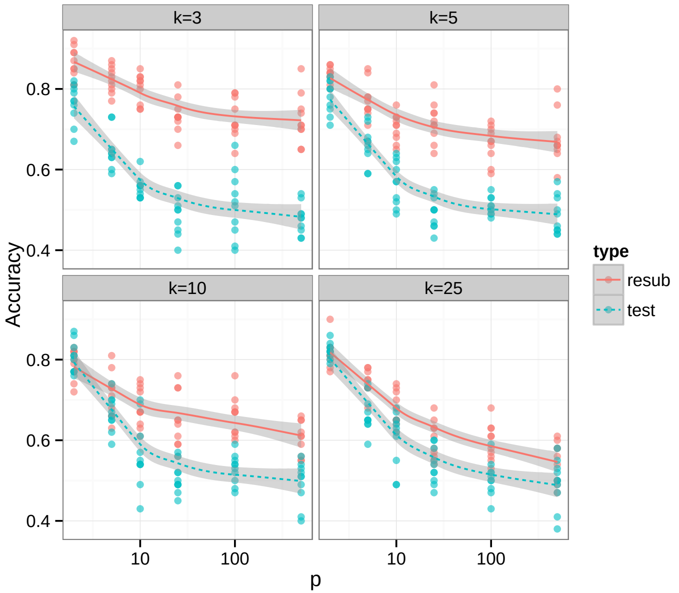
```
# R:

knnTest = knn(
    train = xtrain,
    test = xtest,
    cl = ytrain,
    k = 3
)
nCorrect = sum(diag(table(knnTest, ytest)))
```

```
# Python:

from sklearn.neighbors import KNeighborsClassifier
knnFit = KNeighborsClassifier(n_neighbors=3)
knnFit.fit(array(xtrain), array(ytrain))
knnTest = Series(knnFit.predict(xtest),
                 index = ytest.index)
nCorrect = sum(diag(pandas.crosstab(knnTest, ytest)))
```

k-nearest-neighbors (knn)



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 ...

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 $\mathbf{x}$.

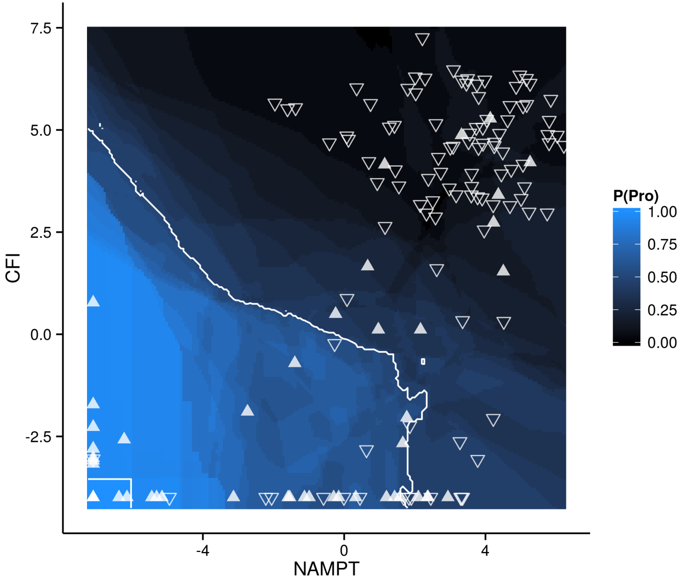
If the dimensionality p is large and r is small, this is very unlikely.

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

Not surprisingly this doesn't always work . . .

May be better to do **feature selection** or **feature extraction** and then fit model using reduced feature set (will return to this idea later).

Overfitting: K=20



Machine Learning Methods for Gene Expression Data

Day 2

Classification

k-Nearest Neighbors

Overfitting

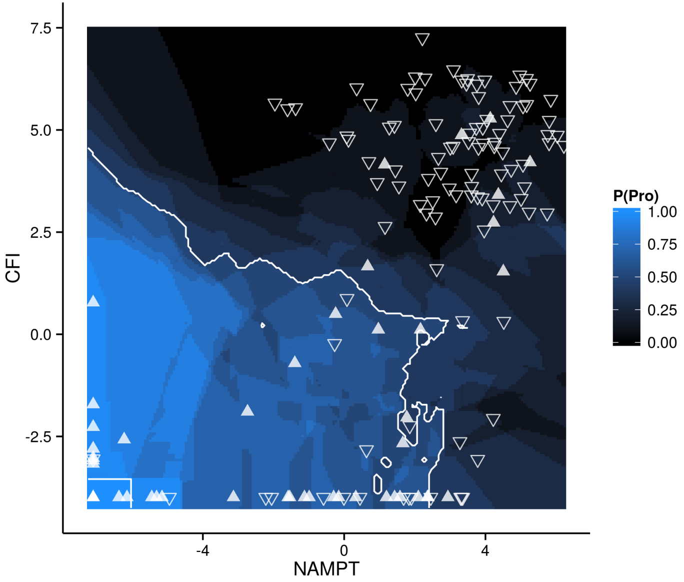
Cross-Validation

Performance Metrics

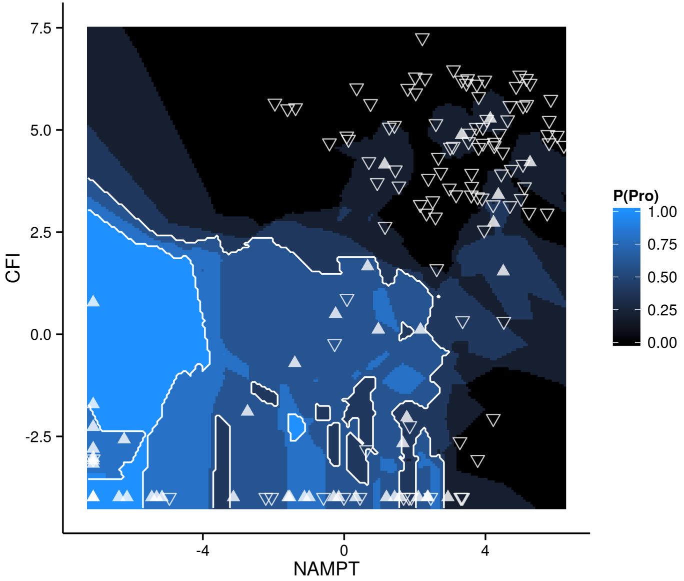
Feature Selection

References

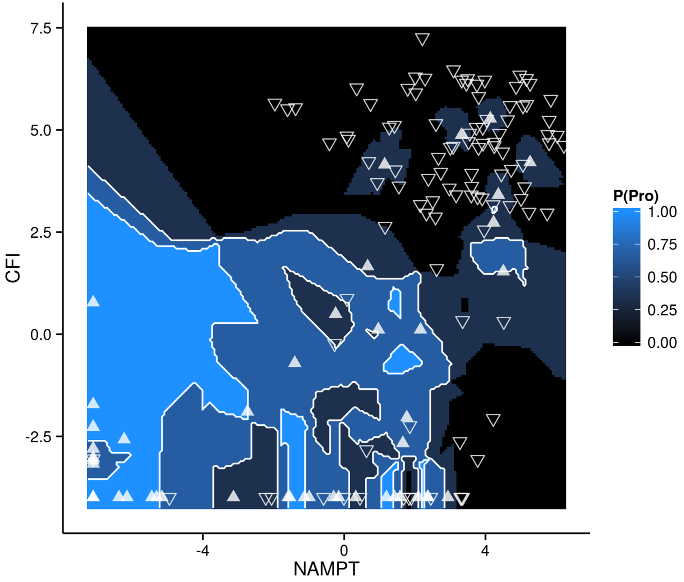
Overfitting: K=10



Overfitting: K=5



Overfitting: K=3



Overfitting: K=1

Machine Learning Methods for Gene Expression Data

Day 2

Classification

k-Nearest Neighbors

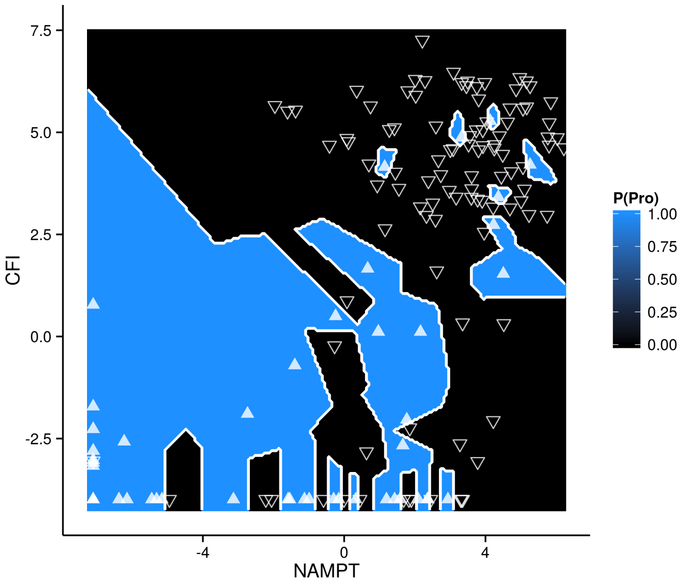
Overfitting

Cross-Validation

Performance Metrics

Feature Selection

References



Cross-Validation (CV)

Have seen that evaluating performance by resubstitution suffers from bias.

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

Cross-Validation (CV)

Have seen that evaluating performance by resubstitution suffers from bias.

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.

Cross-Validation (CV)

Have seen that evaluating performance by resubstitution suffers from bias.

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 $\mathbb{P}_{M,\boldsymbol{\theta}_{\text{-}k}}(Y \mid \mathbf{X} = \mathbf{x}_i)$ are made for samples $i \in S_k$
3. performance estimates are made for each S_k based on $\mathbb{P}_{M,\boldsymbol{\theta}_{\text{-}k}}(Y \mid \mathbf{X} = \mathbf{x}_i)$ and then averaged over all K folds.

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 $\mathbb{P}_{M,\boldsymbol{\theta}_{\text{-}k}}(Y \mid \mathbf{X} = \mathbf{x}_i)$ are made for samples $i \in S_k$
3. performance estimates are made for each S_k based on $\mathbb{P}_{M,\boldsymbol{\theta}_{\text{-}k}}(Y \mid \mathbf{X} = \mathbf{x}_i)$ and then averaged over all K folds.

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.

5-Fold Cross-Validation

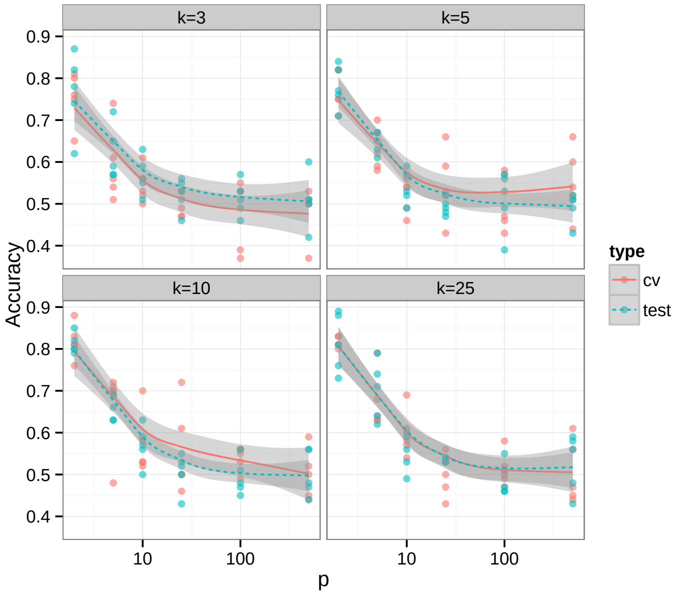
```
# R:

library(caret)
knnCV = train(
    x = xtrain,
    y = ytrain,
    method = "knn",
    tuneGrid = data.frame(k=3),
    trControl = trainControl(method="cv", number=5)
)
cvAccuracyEstimate = knnCV$results[ , "Accuracy"]
```

```
# Python:

from sklearn.neighbors import KNeighborsClassifier
from sklearn.cross_validation import cross_val_score
knnClass = KNeighborsClassifier(n_neighbors=3)
cvAccs = cross_val_score(estimator = knnClass,
                         X = array(xtrain),
                         y = array(ytrain),
                         cv = 5)
cvAccuracyEstimate = mean(cvAccs)
```

5-Fold Cross-Validation



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

Machine Learning Methods for Gene Expression Data

Day 2

Classification

k-Nearest Neighbors

Overfitting

Cross-Validation

Performance Metrics

Feature Selection

References

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)

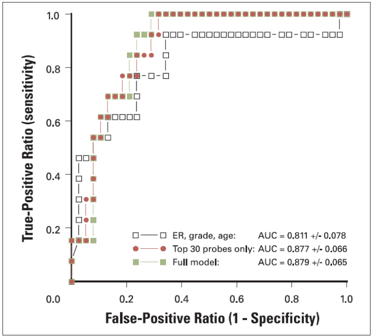


Fig 3. Receiver operating characteristic curves of three distinct pathologic complete response prediction models. The performance of the Diagonal Linear Discriminant Analysis–30 predictor and a predictor based on clinical variables and a combined clinical + pharmacogenomic prediction model are shown 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($y = 0$).

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

Linear Models for Univariate Feature Selection

Most common univariate filter is t-test (or F-test if more than 2 groups) originating from model

$$\mathbb{P}(X_g = x \mid Y = y) = \frac{1}{\sqrt{2\pi\sigma_g^2}} \exp\left[\frac{(x - \mu_{yg})^2}{2\sigma_g^2}\right]$$

Note that we are now considering conditional probabilities of the X_g given Y ...

...and that we are considering the various X_g separately ("univariate" analysis; not necessarily a very good approximation to reality, but at least a tractable one).

Linear Models

Alternately, this model may be described by

$$X_g = \mu_{0g} + (\mu_{1g} - \mu_{0g})Y + \sigma_g \epsilon_g$$

with $\epsilon_g \sim \mathcal{N}(0,1)$, implying

$$\mathbb{E}(X_g \mid Y = y) = \mu_{yg}$$
$$\mathbb{V}(X_g \mid Y = y) = \sigma_g^2$$

Can then rank features based on $|t_g|$ where:

$$t_g = \frac{\hat{\mu}_{0g} - \hat{\mu}_{1g}}{\hat{\sigma}_g \sqrt{\frac{1}{n_0} + \frac{1}{n_1}}}$$

Linear Models

Alternately, this model may be described by

$$X_g = \mu_{0g} + (\mu_{1g} - \mu_{0g})Y + \sigma_g \epsilon_g$$

with $\epsilon_g \sim \mathcal{N}(0, 1)$, implying

$$\mathbb{E}(X_g \mid Y = y) = \mu_{yg}$$
$$\mathbb{V}(X_g \mid Y = y) = \sigma_g^2$$

Can then rank features based on $|t_g|$ where:

$$t_g = \frac{\hat{\mu}_{0g} - \hat{\mu}_{1g}}{\hat{\sigma}_g \sqrt{\frac{1}{n_0} + \frac{1}{n_1}}}$$

While X_g is linear in Y here, it is linearity in μ_{yg} that is generally implied by "linear model" in statistics.

Linear Models

Alternately, this model may be described by

$$X_g = \mu_{0g} + (\mu_{1g} - \mu_{0g})Y + \sigma_g \epsilon_g$$

with $\epsilon_g \sim \mathcal{N}(0, 1)$, implying

$$\mathbb{E}(X_g \mid Y = y) = \mu_{yg}$$
$$\mathbb{V}(X_g \mid Y = y) = \sigma_g^2$$

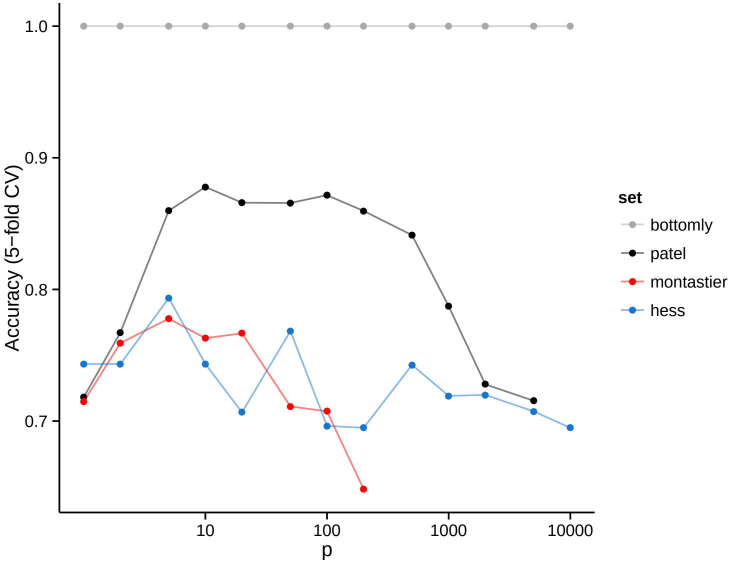
Can then rank features based on $|t_g|$ where:

$$t_g = \frac{\hat{\mu}_{0g} - \hat{\mu}_{1g}}{\hat{\sigma}_g \sqrt{\frac{1}{n_0} + \frac{1}{n_1}}}$$

While X_g is linear in Y here, it is linearity in μ_{yg} that is generally implied by "linear model" in statistics.

We will see later that with a few modifications linear models can be good candidates for modeling Y based on $\mathbf{X}$ too.

knn Accuracy With t-Test Feature Selection



References I

Machine Learning Methods for Gene Expression Data

Day 2

Classification

k-Nearest Neighbors

Overfitting

Cross-Validation

Performance Metrics

Feature Selection

References

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.

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.

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