Adaptive Resampling in a Parallel World

Max Kuhn

Pfizer Global R&D
Nonclinical Statistics
Groton

July 2, 2014

Tuning Parameters and Predictive Models

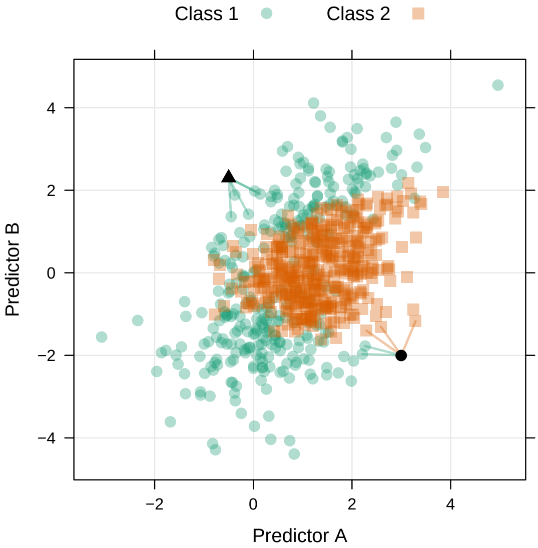
Some machine learning/predictive models have specific "knobs" to control over-fitting

- neighborhood size in nearest neighbor models is an example
- the number if splits in a tree model

Often, poor choices for these parameters can result in over-fitting and there may not be an analytical method for estimating their value.

For example, the next slide shows a data set with two predictors. We want to be able to produce a line (i.e. decision boundary) that differentiates two classes of data.

5–Nearest Neighbor Models



Grid Search and Resampling

The typical way one might determine appropriate values of the tuning parameter(s) is to use some form of resampling, e.g. cross–validation or the bootstrap.

Resampling can take the training set data and get reasonable assessments of how well the model would work on future samples.

The user could define a candidate set of tuning parameter combinations and evaluate each using resampling, i.e. a grid search.

Grid Search Resampling Algorithm

$D =$	Training set
$B =$	Number of resamples
$i =$	Resampling iteration i
$R_i =$	Resampled data set
$T_i =$	Resampled holdout set
$p =$	Number of tuning settings
$\Theta =$	Tuning parameter grid
$\theta_j =$	Tuning parameter value
$Q_{ij} =$	Performance estimate
$\hat{f}(D;\theta) =$	Trained model

```
Define parameter set $\Theta$;
for $i = 1 \dots B$ do
    Generate $R_i$ and $T_i$;
    for $j = 1 \dots p$ do
        Fit $\hat{f}_{ij}(R_i; \theta_p)$;
        Predict $T_i$ to estimate $Q_{ij}$;
    end
end
Calculate $\hat{Q}_1 \dots \hat{Q}_p$;
Determine $\theta_{opt}$;
Fit the final model $\hat{f}(D; \theta_{opt})$;
```

Mutagenicity Data

Kazius et al. (2005) created a data set that attempts to predict the potential for a chemical to cause *mutagenicity*.

They labeled 4,335 compounds as either mutagen or non–mutagen and we generated 830 descriptors of molecular structure for each compound as predictors.

Examples of the descriptors used in these analyses are atom counts, molecular weight, surface area and other measures of size and charge.

Using a predictive model, future compounds can be assessed for their potential toxicity based on these properties.

Mutagenicity Data

A support vector machine (SVM) model using a radial basis function was used to predict mutagenicity.

The method of Caputo et al. (2002) was used to estimate the radial basis function kernel parameter σ analytically. so that we only tune over the SVM cost parameter.

Cost was varied over 21 values: $\Theta = \left\{2^{-2}, 2^{-1.5} \ldots, 2^{8}\right\}$

Simple bootstrap resampling was used to tune the model with $B = 50$.

Using "pick–the–winner", the optimal value was estimated to be $\theta_{opt} = 2^{1.5}$.

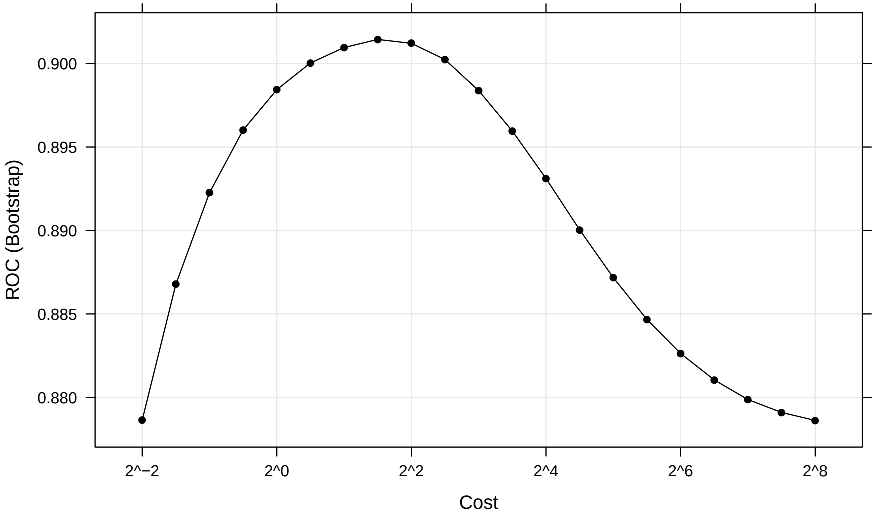
Basic Resampling using caret

```r
library(caret)

ctrl <- trainControl(method = "boot", number = 50)

fit <- train(x = predictors, y = outcome,
             method = "svmRadial",
             preProc = c("center", "scale"),
             tuneLength = 21,
             trControl = ctrl)
```

Model Tuning Results



Futility Analysis

Why should we wait until all $B \times p = 1,050$ models are fit before we eliminate parts of Θ?

We might consider a tuning parameter as futile if it is unlikely to have optimal performance.

At some point during resampling (B_{min}), we can look at the data to potentially discard candidate values of Θ.

We extend the work of Shen et al. (2011) to analyze the estimated Q_{ij}. A manuscript is also in review that also includes a novel method of assessing futility.

Generalized Linear Models for Measuring Futility

The resampled performance estimates are analyzed using a generalized linear model:

$$Q_{jk} = \mu + \tau_j + \epsilon_{ik}$$

where $\epsilon_{ik} \sim N(0, \Sigma)$ with $\Sigma_k = \sigma^2(1-\rho)I_{p_i} + \sigma^2\rho J_{p_i}$.

The grand mean μ is associated with the current best tuning parameter at iteration k so that the τ_j estimate the loss of performance for setting j.

To estimate futility, a one–sided upper confidence interval for the $\widehat{\tau}_j$ are constructed.

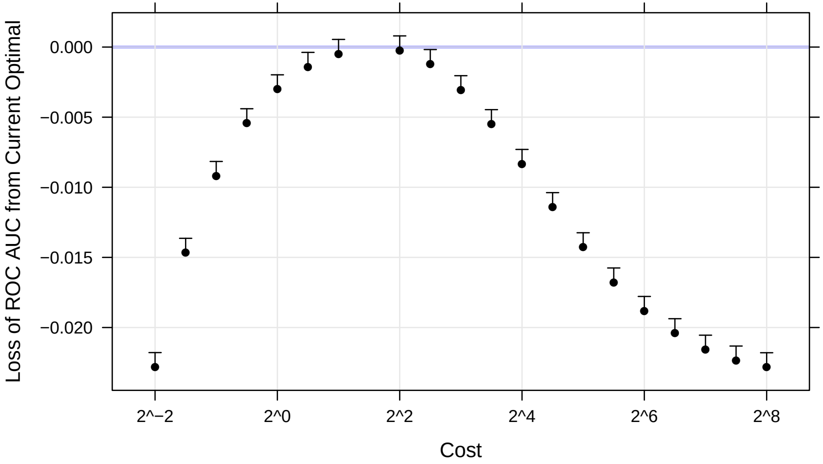
If the interval includes zero, setting j is resampled on the next iteration.

Adaptive Resampling Algorithm

```
Define parameter set $\Theta$;
for $i = 1 \ldots B$ do
    Generate $R_i$ and $T_i$;
    for $j = 1 \ldots p_i$ do
        Fit $\hat{f}_{ij}(R_i; \theta_{p_i})$;
        Predict $T_i$ to estimate $Q_{ij}$;
    end
    if $i > B_{min}$ and $p_i > 1$ then
        Calculate $\hat{Q}_1 \ldots \hat{Q}_{p_i}$;
        Conduct futility analysis;
        Remove parameter settings;
        Update $p_i$;
        if $p_i = 1$ then set $\theta_{opt}$;
    end
end
if $p_i > 1$ then Determine $\theta_{opt}$;
Fit the final model $\hat{f}(D; \theta_{opt})$;
```

Using this procedure, the same optimal settings were found by computing only 28.5% of the possible models resulting in a **speed–up of 3.5**.

At $B_{min} = 5$ and $\alpha = 0.05$



Basic Resampling using caret

```r
library(caret)

ctrl <- trainControl(method = "boot", number = 50)

fit <- train(x = predictors, y = outcome,
             method = "svmRadial",
             preProc = c("center", "scale"),
             tuneLength = 21,
             trControl = ctrl)
```

Adaptive Resampling using caret

```
library(caret)

ctrl <- trainControl(method = "adaptive_boot", number = 50)

fit <- train(x = predictors, y = outcome,
             method = "svmRadial",
             preProc = c("center", "scale"),
             tuneLength = 21,
             trControl = ctrl)
```

The "Sub–Model" Trick

There are some models in R where we don't have to execute the model function for each possible sub–model (e.g. `gbm`, `plsr`).

The current complete set includes `model %in% c("bagEarth", "blackboost", "bstLs", "bstSm", "bstTree", "C5.0", "C5.0Cost", "cubist", "earth", "enet", "foba", "gamboost", "gbm", "glmboost", "glmnet", "kernelpls", "lars", "lars2", "lasso", "lda2", "leapBackward", "leapForward", "leapSeq", "LogitBoost", "pam", "partDSA", "pcr", "PenalizedLDA", "pls", "relaxo", "rpart", "rpart2", "rpartCost", "simpls", "superpc", "widekernelpls")`

So far, timing tests still show improvments in many of these models by using adaptive resampling.

Parallel Processing

How does parallel processing affect this process?

Does it obviate the gains one would get from adaptive resampling?

When $i < B_{min}$ or $p_i = 1$, resamples can be run parallel.

Within a given resample, surviving models can always be run in parallel.

```
Define parameter set $\Theta$;
for $i = 1 \ldots B$ do
    Generate $R_i$ and $T_i$;
    for $j = 1 \ldots p_i$ do
        Fit $\hat{f}_{ij}(R_i; \theta_{p_i})$;
        Predict $T_i$ to estimate $Q_{ij}$;
    end
    if $i > B_{min}$ and $p_i > 1$ then
        Calculate $\hat{Q}_1 \ldots \hat{Q}_{p_i}$;
        Conduct futility analysis;
        Remove parameter settings;
        Update $p_i$;
        if $p_i = 1$ then set $\theta_{opt}$;
    end
end
if $p_i > 1$ then Determine $\theta_{opt}$;
Fit the final model $\hat{f}(D; \theta_{opt})$;
```

Adaptive Resampling in Parallel

Resampling	Method	Time (hr)
Complete	Sequential	40.5
Adaptive	Sequential	11.6
Complete	Parallel	14.2
Adaptive	Parallel	3.6

Adaptive only speed-up: 3.5

Parallel only speed-up: 2.9

Adaptive and parallel speed–up: 11.3

Adaptive Resampling using caret Sequentially (again)

```
library(caret)

ctrl <- trainControl(method = "adaptive_boot", number = 50)

fit <- train(x = predictors, y = outcome,
             method = "svmRadial",
             preProc = c("center", "scale"),
             tuneLength = 21,
             trControl = ctrl)
```

Adaptive Resampling using caret in Parallel

```r
library(doMC) ## or whatever "do" package you like
registerDoMC(cores = 6)

library(caret)

ctrl <- trainControl(method = "adaptive_boot", number = 50)

fit <- train(x = predictors, y = outcome,
             method = "svmRadial",
             preProc = c("center", "scale"),
             tuneLength = 21,
             trControl = ctrl)
```

Summary

If the training set size is big enough, adaptive resampling can generate quality models.

If the computationally complexity is large, it can also generate significant speed–ups

Parallel processing does not obviate the gains generated from adaptive resampling

In the short term, this code will be included in the `caret` package.

The package contains another method for determining futility based on a Bradley–Terry model.

Acknowledgments

Thanks for Kjell Johnson and David Potter for reviewing the manuscript that lead to this presentation.

References

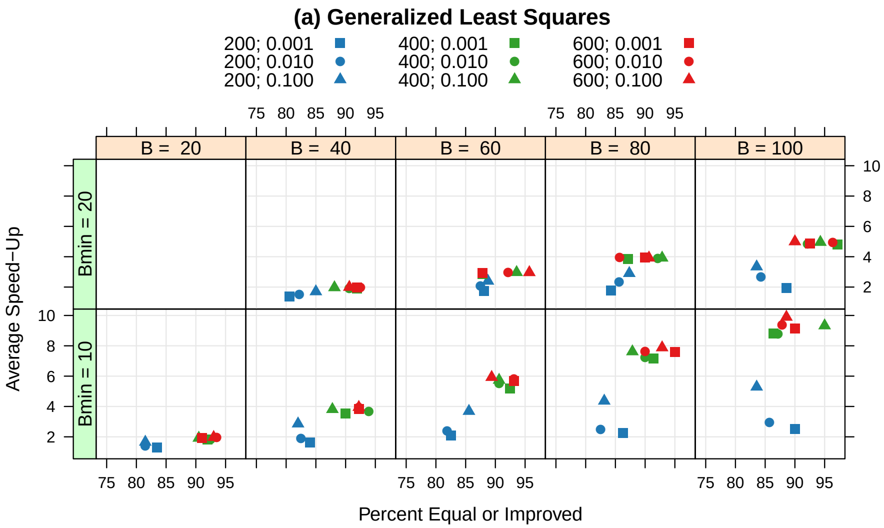
Caputo B, Sim K, Furesjo F, Smola A (2002). Appearance–based object recognition using SVMs: Which kernel should I use? in *Proceedings of NIPS Workshop on Statistical Methods for Computational Experiments in Visual Processing and Computer Vision*.

Kazius, J., McGuire, R., & Bursi, R. (2005). Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1), 312–320.

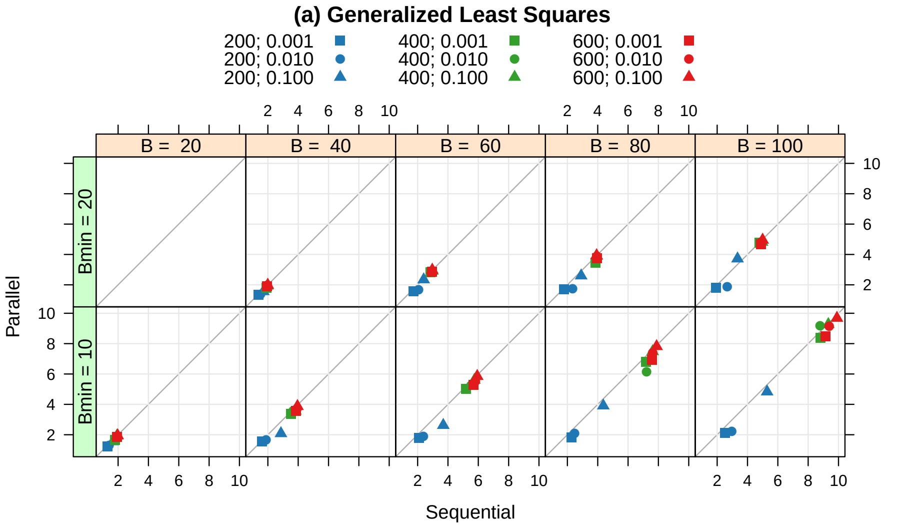
Kuhn, M. (2014). **Futility analysis in the cross–validation of machine learning models**. *arXiv* arXiv:1405.6974.

Shen, H., Welch, W. J., & Hughes-Oliver, J. M. (2011). Efficient, adaptive cross-validation for tuning and comparing models, with application to drug discovery. *The Annals of Applied Statistics*, 5(4), 2668–2687.

Does It Work? Simulation Results



More Simulation Results


(a) Generalized Least Squares
200; 0.001
200; 0.010
200; 0.100
400; 0.001
400; 0.010
400; 0.100
600; 0.001
600; 0.010
600; 0.100
B = 20
B = 40
B = 60
B = 80
B = 100
Bmin = 20
Bmin = 10
Parallel
Sequential