MA8701 Advanced methods in statistical inference and learning

L8: Hyperparameter tuning with Bayesian Optimization and Evaluating and comparing results from prediction models

Mette Langaas IMF/NTNU

28 February, 2021

Part 4 - final act

Outline

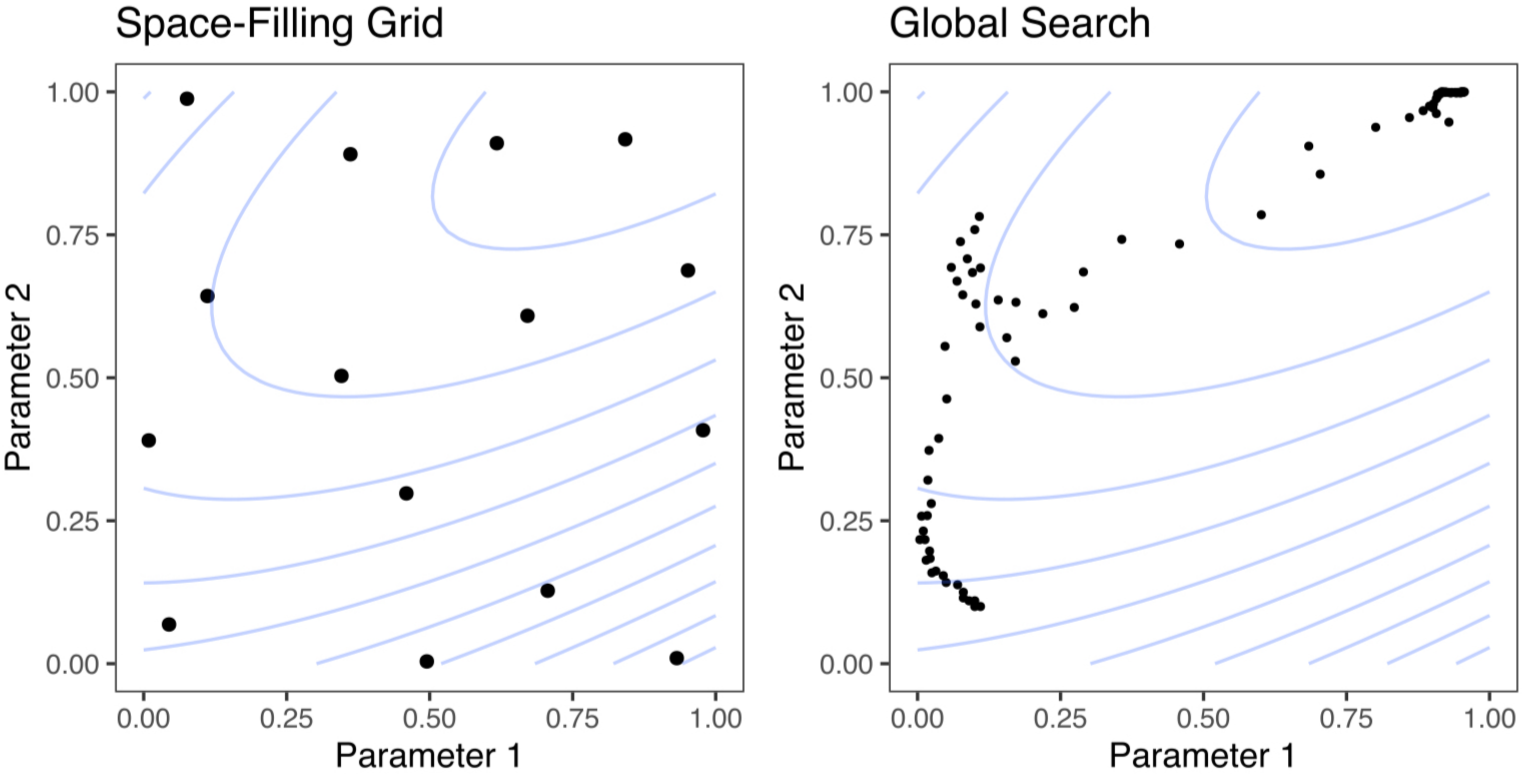
- Hyperparameter tuning - with Bayesian Optimization
- Evaluating and comparing results from prediction models

Choosing hyperparameters

- What are *hyperparameters*?
- Which hyperparameters have we encountered in the course so far?

Kuhn & Silge (2021)

Grid search vs iterative search



Surrogate methods

We will look at two types of surrogate models: Bayesian regression with Gaussian processes (in Bayesian optimization) and regression-type models in response surface methods (presented by group 2).

Design of experiments and response surface methodology

Article presentation by group 2.
G. A. Lujan-Moreno, P. R. Howard, O. G. Rojas and D. C. Montgomery (2018): Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case- study. Expert Systems with Applications. 109, 195-205.

Bayesian optimization

Bayesian optimization is an iterative method - where we start with evaluating some loss function at some predefined set of points in the hyperparameter space. New position in the hyperparameter space are chosen iteratively.

Two key ingredients:

- a surrogate model (we will only look at Bayesian regression with Gaussian processes) to fit to the observed values of the loss function in the hyperparameter space
- an *acquisition* function to decide a new point in the hyperparameter space to evaluate next

Underlying idea: given some “observations” in the hyperparameter space, the task is to decide where to place a new point. We should try a point where:

- we expect a good value and/or
- we have little information so far

To do that we need information on both expected value *and* variance - or preferably the distribution of the loss function for your problem.

Gaussian processes

(Eidsvik 2017, page 6-7, note in TMA4265)

A Gaussian process is defined for

- times or locations x_i, $i = 1, \dots, n$ in $\Re^d$, where
- $Y_i = Y(x_i)$ is a random variable at x_i
- such that $\mathbf{Y} = (Y_1, \dots, Y_n)$ is multivariate Gaussian.

The process is *first order (mean) stationary* if $\mathrm{E}(Y(x)) = \mu$ for all x, and this can be extended to depend on covariates.

The process is *second order stationary* if $\mathrm{Var}(Y(t)) = \sigma^2$ for all x and the correlation $\mathrm{Corr}(Y(x), Y(x'))$ only depends on differences between x and x'.

The multivariate Gaussian distribution is defined by the mean and covariance alone.

Correlation functions

- We assume that points at positions close to each other have a stronger correlation than point far apart.

Power exponential or Gaussian kernel

$$\text{Corr}(Y(x), Y(x')) = \exp(-\phi_G \|x - x'\|^2)$$

where the L2 distance is used and ϕ_G is a parameter that determine the decay in the correlations.

Matern-type kernel

$$\text{Corr}(Y(x), Y(x\acute{}\,)) = (1 + \phi_M \|x - x'\|) \exp(-\phi_M \|x - x'\|)$$

decay-describing parameter ϕ_M.

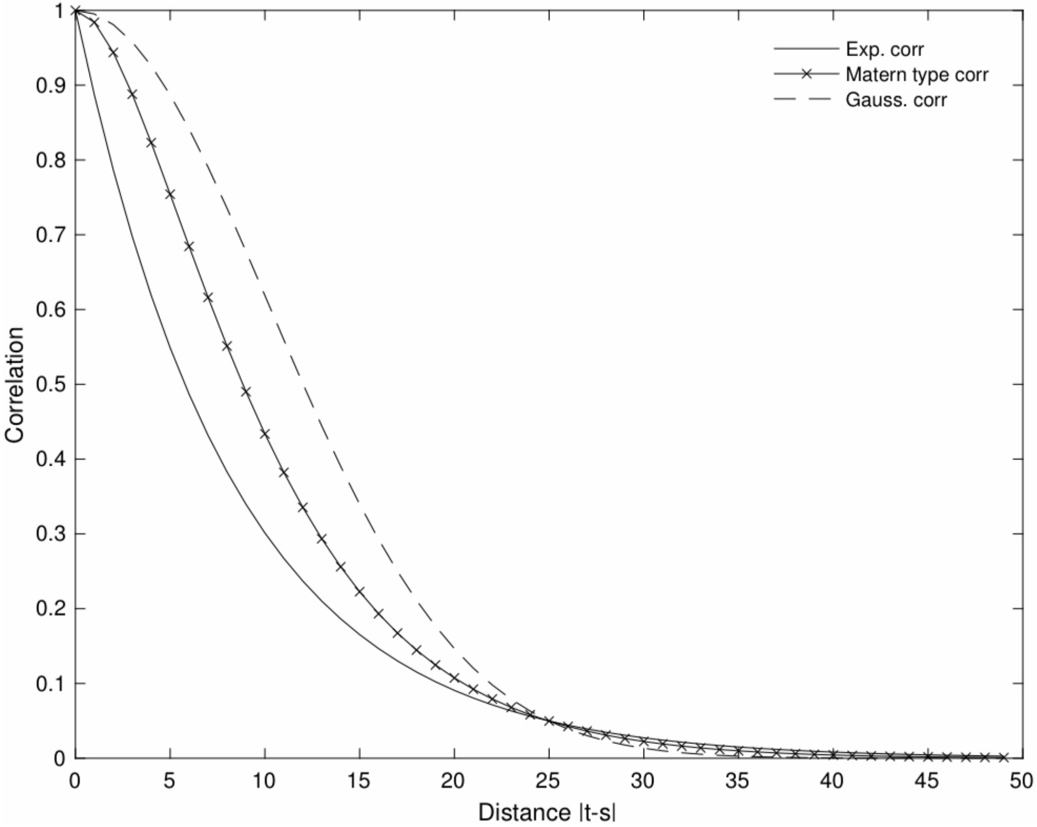


Figure 4: Three different correlation functions. Eidsvik (2017)

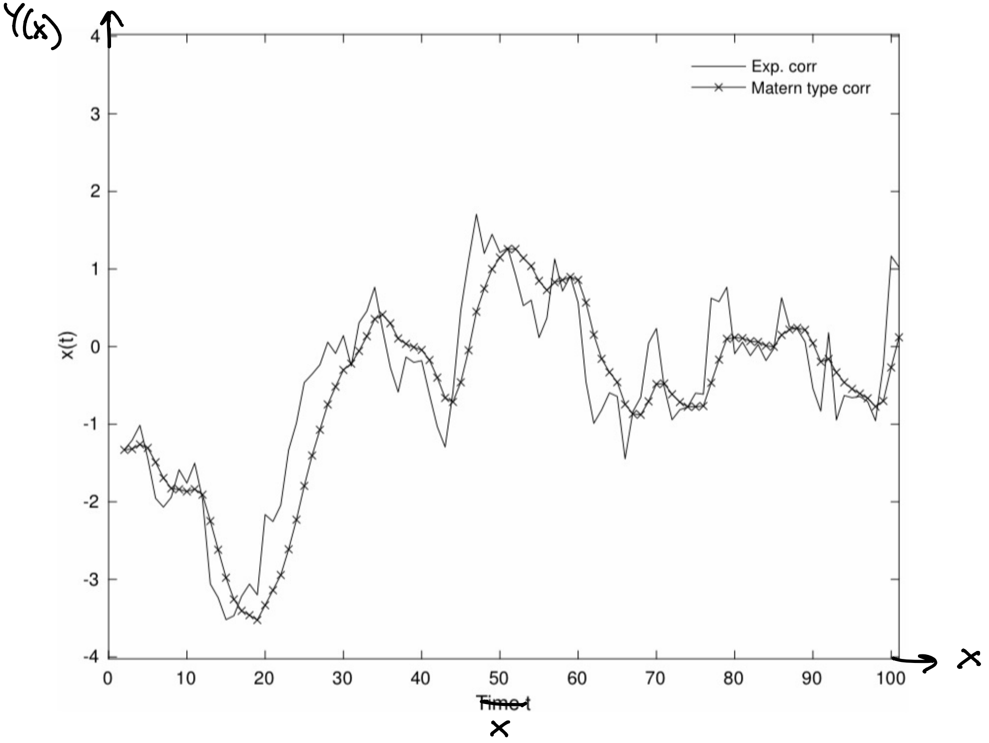


Figure 5: One realization from the Gaussian process with exponential covariance function and one with Matern type correlation function. The mean is 0 and variance 1. The correlation decay parameters are $\phi_E = 3/25$ and $\phi_M = 0.19$.

Eidvik (2017)

From correlations into covariance matrix

For simplicty assume that $d = 1$. The number of positions to consider is n.

To get from correlation function to a $n \times n$ covariance matrix first construct a $n \times n$ matrix of distances for each pair of positions, denote this $\mathbf{H}$.

For the Matern-type correlation function the covariance matrix can then be written

$$\Sigma = \sigma^2(1 + \phi_M \mathbf{H}) \otimes \exp(-\phi_M \mathbf{H}))$$

where $\otimes$ is elementwise multiplication.

Multivariate normal distribution

The random vector $\mathbf{Y}_{p\times 1}$ is multivariate normal N_p with mean and (positive definite) covariate matrix Σ. The pdf is:

$$f(\mathbf{Y}) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\mathbf{Y} - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{Y} - \boldsymbol{\mu})\}$$

The conditional distributions of the components are (multivariate) normal.

$$\mathbf{Y}_2 \mid (\mathbf{Y}_1 = \mathbf{Y}_1) \sim N_{p2}(\boldsymbol{\mu}_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{Y}_1 - \boldsymbol{\mu}_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}).$$

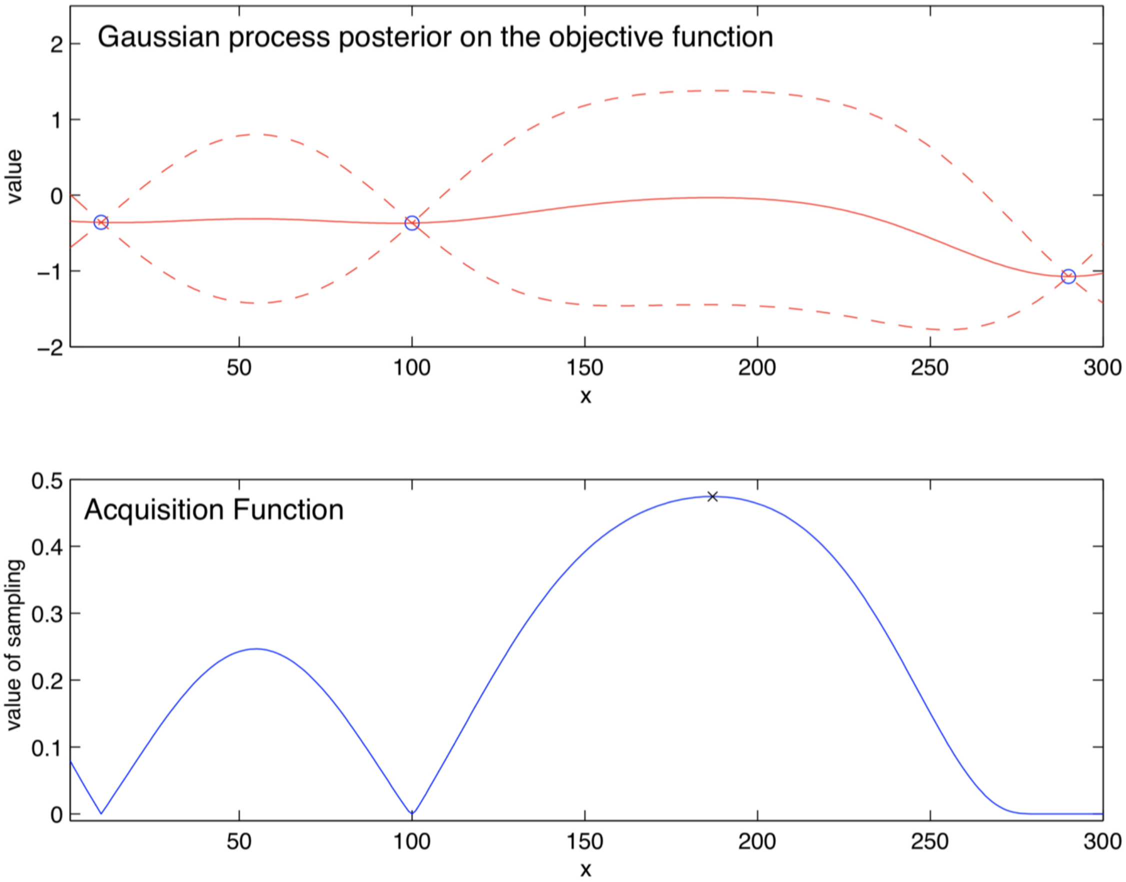


Figure 1: Illustration of BayesOpt, maximizing an objective function f with a 1-dimensional continuous input. The top panel shows: noise-free observations of the objective function f at 3 points, in blue; an estimate of $f(x)$ (solid red line); and Bayesian credible intervals (similar to confidence intervals) for $f(x)$ (dashed red line). These estimates and credible intervals are obtained using GP regression. The bottom panel shows the acquisition function. Bayesian optimization chooses to sample next at the point that maximizes the acquisition function, indicated here with an "x."

Frazier (2018)

Acquisition function: Expected improvement

(Frazier 2018 page 7)
Thought experiment:

1) we have evaluated our function at all possible points x, and must return a solution based on what we already have evaluated. If the evaluation is noise-less we need to return the point with the largest observed value f.
2) Correction: We may perform one more evaluation. If we choose x we observe $f(x)$, and the best point before that was f_n^*. The improvement at the new observation is then

$$\max(f(x) - f_n^*, 0)$$

3) We define the *expected improvement* as

$$\mathsf{EI}_n(x) = \mathsf{E}_n[\max(f(x) - f_n^*, 0)]$$

where the expectation is taken at the posterior distribution given that we have evaluated f at n observations $x_1, \ldots, x_n$, and the posterior distribution is that f conditional on $x_1, \ldots, x_n, y_1, \ldots, y_n$ is normal with mean $\mu_n(x)$ and variance $\sigma_n^2(x)$.

4) How to evaluate the expected improvement? Integration by parts gives

$$\mathsf{EI}_n(x) = \max(\mu_n(x) - f_n^*, 0)]) + \sigma_n(x)\phi(\frac{\max(\mu_n(x) - f_n^*, 0)}{\sigma_n(x)})$$

$$-\mathsf{abs}(\mu_n(x) - f_n^*)\Phi(\frac{\max(\mu_n(x) - f_n^*, 0)}{\sigma_n(x)})$$

$\mu_n(x) - f_n^*$ is expected proposed vs previously best

5) We choose to evaluate the point with the largest expected improvement

$$x_{n+1} = \mathsf{argmaxEI}_n(x)$$

Frazier (2018)

```
Place a Gaussian process prior on $f$.
Observe $f$ at $n_0$ points from some experimental design. Set $n = n_0$.
while $n \leq N$ do
Update the posterior on f with all available data
Let $x_n$ be a maximizer of the acquisition function over $x$, computed using the current posterior
Observe $y_n = f(x_n)$
Increment $n$
end while
Return a solution: a point with largest $f(x)$ or the point with the largest posterior mean
```

Extension

What is the objection function is not observed noise-less? Independent normal error term ε can be added to the previously defined $Y = f(x)$ to make a new $Y = f(x) + \varepsilon$. This (only) adds a diagonal term to the covariance matrix, and it is common to assume that the variance is the same for all x and treat the variance as a hyperparameter.

The R the function **tune_bayes** is available in the package **tune**, and requires tha
a workflow. Default in the GP is exponential correlation function, but first we tr

```
tree_rec <- recipe(medv~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+pt

tune_spec <- rand_forest( # parsnip interface to random forests mode
  mode="regression",
  mtry = tune(),
  trees = tune(),
#  min_n = tune()
) %>%
#  set_mode("regression") %>%
#  set_engine("ranger",objective="reg:rmse") # errors with ranger
  set_engine("randomForest") # randomforest ok

tune_wf <- workflow() %>%
  add_recipe(tree_rec) %>%
  add_model(tune_spec)

tune_param <- tune_spec%>%
  parameters%>%
  update(mtry=mtry(c(1L,13L)),trees=trees(c(100L,500L)))
```

```
vfold  <- vfold_cv(Boston, v = 5)
# then trying BO
ctrl <- control_bayes(verbose = TRUE)
bayesres<- tune_bayes(tune_wf,
    resamples = vfold,
    #metrics = rmse,
    corr=list(type="matern",nu=5/2),
    #default in corr_mat(GPfit) is "exponential" power 1.95
    initial = 10,
    param_info = tune_param,
    iter = 10,
    objective=exp_improve(),
    control = ctrl
  )
dput(bayesres,"bayesres.dd")
```

```
## # A tibble: 10 x 9
##     mtry trees .metric .estimator  mean     n std_err .config              .iter
##    <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>                <int>
##  1     6   204 rmse    standard    3.16     5   0.231 Iter6                    6
##  2     7   315 rmse    standard    3.17     5   0.257 Iter10                  10
##  3     6   319 rmse    standard    3.19     5   0.254 Iter7                    7
##  4     6   210 rmse    standard    3.19     5   0.262 Iter2                    2
##  5     6   196 rmse    standard    3.20     5   0.252 Preprocessor1_Model~     0
##  6     6   296 rmse    standard    3.22     5   0.256 Preprocessor1_Model~     0
##  7     7   204 rmse    standard    3.22     5   0.287 Iter5                    5
##  8     8   305 rmse    standard    3.23     5   0.280 Preprocessor1_Model~     0
##  9     7   333 rmse    standard    3.23     5   0.271 Iter8                    8
## 10     9   452 rmse    standard    3.24     5   0.283 Preprocessor1_Model~     0
```

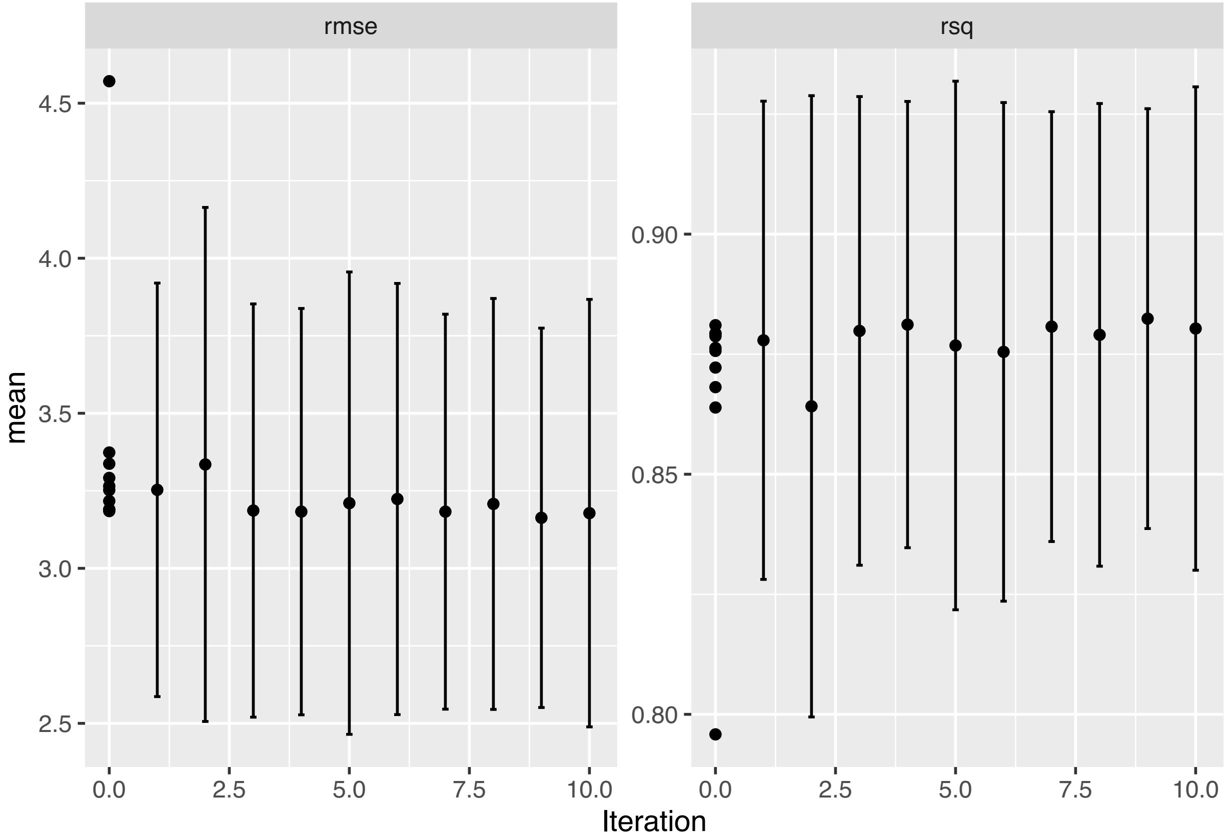
Here we try the default exponential correlation.

```
bayesres2<- tune_bayes(tune_wf,
    resamples = vfold,
    #metrics = rmse,
    #corr=list(type="matern",nu=5/2),
    #default in corr_mat(GPfit) is "exponential" power 1.95
    initial = 10,
    param_info = tune_param,
    iter = 10,
    objective=exp_improve(),
    control = ctrl
  )
dput(bayesres2,"bayesres2.dd")
```

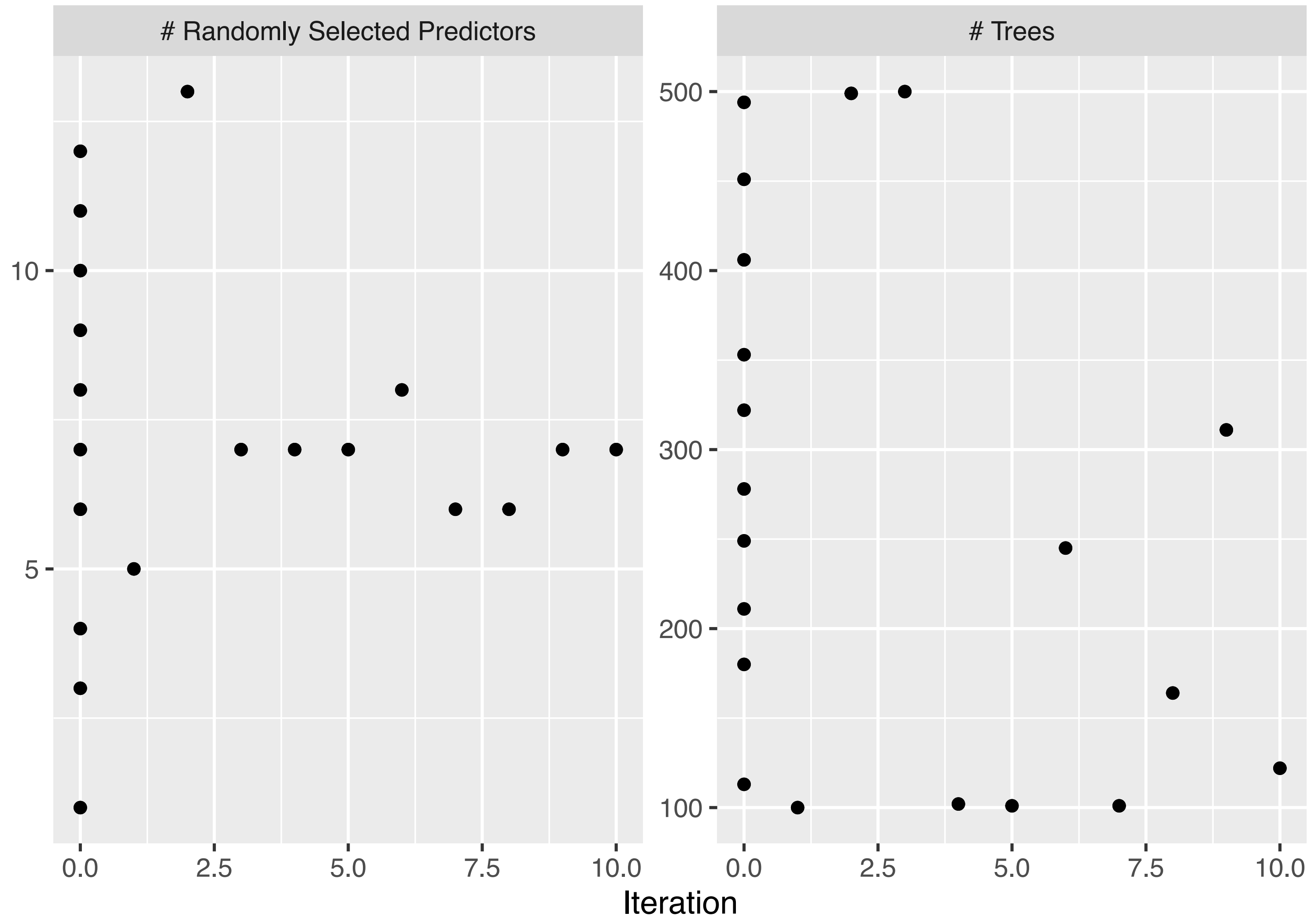
```
bayesres2=dget("bayesres2.dd")
show_best(bayesres2,n=10)
```

```
## # A tibble: 10 x 9
##     mtry trees .metric .estimator  mean     n std_err .config              .iter
##    <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>                <int>
##  1     7   311 rmse    standard    3.16     5   0.238 Iter9                    9
##  2     7   122 rmse    standard    3.18     5   0.268 Iter10                  10
##  3     6   101 rmse    standard    3.18     5   0.248 Iter7                    7
##  4     7   102 rmse    standard    3.18     5   0.255 Iter4                    4
##  5     7   249 rmse    standard    3.18     5   0.270 Preprocessor1_Model~     0
##  6     7   500 rmse    standard    3.19     5   0.259 Iter3                    3
##  7     6   278 rmse    standard    3.19     5   0.255 Preprocessor1_Model~     0
##  8     8   113 rmse    standard    3.19     5   0.257 Preprocessor1_Model~     0
##  9     6   164 rmse    standard    3.21     5   0.258 Iter8                    8
## 10     7   101 rmse    standard    3.21     5   0.290 Iter5                    5
```

```
autoplot(bayesres2,type="performance")
```



```
autoplot(bayesres2,type="parameters")
```



# Randomly Selected Predictors
# Trees
10
5
500
400
300
200
100
0.0
2.5
5.0
7.5
10.0
Iteration

Suggested software

- R: DiceOptim (on CRAN)
- R: tune_bayes in `tune` (also CRAN)
- Python: Spearmint https://github.com/HIPS/Spearmint
- Python: GPyOpt https://github.com/SheffieldML/GPyOpt
- Python: GPFlow (Tensorflow) https://github.com/GPflow/GPflow and GPYTorch (PyTorch) https://github.com/cornellius-gp/gpytorch

Group task

Construct a concept map for Bayesian Optimization or hyperparameter tuning.

Show the map using your camera or by sharing screen!

Evaluating and comparing results from prediction models

We will only consider using one data set. For comparing methods across many data sets see Boulesteix et al (2015).

We are not interesting in general “unconditional” results (for all possible training sets from some distribution) - and not to know if method A *in general* is better than method B in situations similar to ours.

We also have the “No free lunch theorem” of Wolpert (1996) stating that there is no such thing as the “best” learning algorithm.

We consider two different situations.

Data rich situation:

- We have used our *training set* to tune our model (choosing hyperparameters) - possibly by using cross-validation or some other technique.
- Then we have fitted the finally chosen model to the full training set, and used this final model to make predictions on the *test set*.
- If we want to compare results from two or more prediction models, when the same test set is used for all the models.

Data poor situation:

- We don´t have enough data to set aside observations for a test set.
- We need to use some type of resampling to evaluate and compare prediction models.
- This is more difficult than for the data rich situation, because now *independence* of observations for testing cannot be assumed (more below).

What do we want?

Classification

- Estimate and confidence interval for misclassification rate (or ROC-AUC) on test observations for one prediction model.
- Is the misclassification rate (or ROC-AUC) for prediction method A better than for prediction method B? Can this be extended to more than two methods?

Far the most popular situation in the literature.

Regression

Relate to ELS Ch7.1 with Err and $\text{Err}_{\mathcal{T}}$.

- Estimate and confidence interval for evaluation criterion (mean square error of predictions) on test observations for one prediction model.
- Is prediction model A better than prediction model B? Can this be extended to more than two methods?

Much more difficult to “find” literature with methods here than for classification - seems to be far less popular.

Keep in mind that not only error rates govern which prediction models to use, also aspects like training time and interpretability plays an important role.

There might be controllable and uncontrollable factors that influence the model fit and add variability to our model predictions.

It is always wise to present results in graphical displays.

Data rich situation

Assumptions:

- Both the training set (size N) and the test set (size M) are drawn as random samples from the population under study, and are independent of each other.
- The training set is used to estimate (one or many) prediction model(s),
- and predictions are provided (for each prediction method) for the M observations in the test set.
- The M predictions $\hat{y}_i$, $i = 1, \ldots, M$ are independent.
- If we have predictions from two methods A and B, these are made on the same test observations, and the triplets $(y_i, \hat{y}_i^A, \hat{y}_i^B)$ are independent for $i = 1, \ldots, M$.

Classification

Example

We will use the classical data set of *diabetes* from a population of women of Pima Indian heritage in the US, available in the R `MASS` package. The following information is available for each woman:

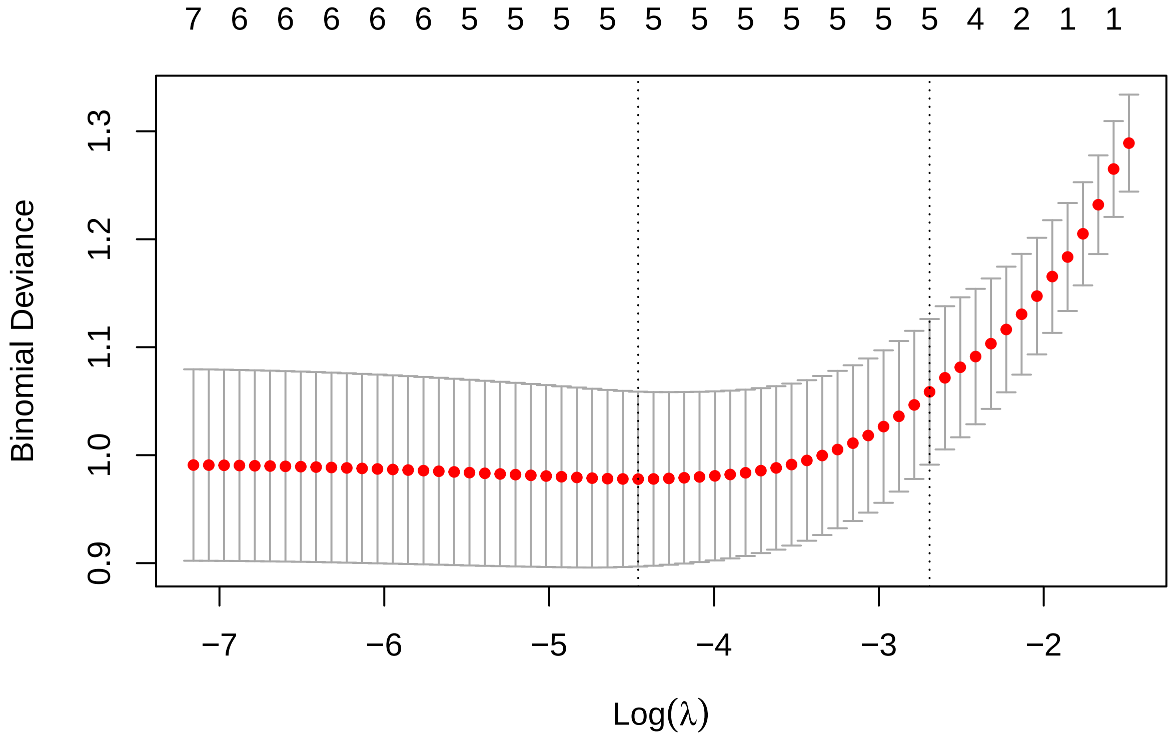
- diabetes: 0= not present, 1= present
- npreg: number of pregnancies
- glu: plasma glucose concentration in an oral glucose tolerance test
- bp: diastolic blood pressure (mmHg)
- skin: triceps skin fold thickness (mm)
- bmi: body mass index (weight in kg/(height in m)^2)
- ped: diabetes pedigree function.
- age: age in years

We will use the default division into training and test in the MASS library, with 200 observations for training and 332 for testing.

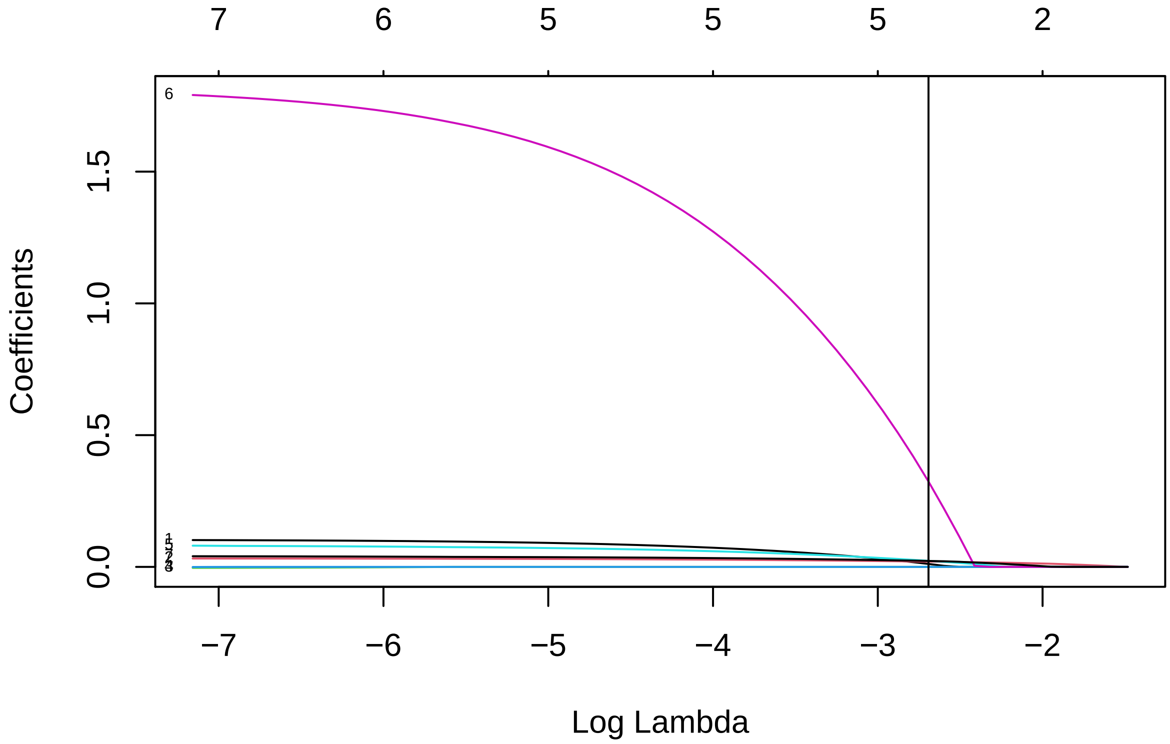
```r
Pima.tr$diabetes=as.numeric(Pima.tr$type)-1
Pima.te$diabetes=as.numeric(Pima.te$type)-1
train=Pima.tr[,c(1:7,9)]
test=Pima.te[,c(1:7,9)]
colnames(test)=colnames(Pima.te)[c(1:7,9)]

# logistic lasso
xs=model.matrix(~.-diabetes,data=train)[,-1]
xstest=model.matrix(~.-diabetes,data=test)[,-1]
fitlasso=glmnet(xs,train$diabetes,family=binomial(link="log
cvlasso=cv.glmnet(x=xs,y=train$diabetes,family="binomial")
plot(cvlasso)
```

```
fitlasso=glmnet(xs,train$diabetes,family=binomial(link="logit"))
cvlasso=cv.glmnet(x=xs,y=train$diabetes,family="binomial")
plot(cvlasso)
```



```
plot(fitlasso,xvar="lambda",label=TRUE);
abline(v=log(cvlasso$lambda.1se));
```



```
predlasso=predict(fitlasso,newx=xstest,s=cvlasso$lambda.1se,type="response")
classlasso=ifelse(predlasso > 0.5, 1, 0)
table(test$diabetes, classlasso)# 223 non-diabetes and 109 diabetes cases
```

```
##    classlasso
##       0   1
##   0 213  10
##   1  61  48
```

```
# randomforest
rf=randomForest(factor(diabetes)~.,data=train,mtry=2,ntree=500,importance=TRUE)
rf$confusion #error rates based on OOB data
```

```
##     0  1 class.error
## 0 111 21   0.1590909
## 1  35 33   0.5147059
```

```
classrf=predict(rf,newdata=test)
table(test$diabetes, classrf)# 223 non-diabetes and 109 diabetes cases
```

```
##    classrf
##       0   1
##   0 193  30
##   1  45  64
```

```
rfpred = predict(rf,test, type = "prob")[,2]
```

Binomial CI

A common evaluation criterion is the misclassification rate.
Let p be the probability of success (correct classification) for a prediction method. In our test set we have M independent observations, with associated predictions $\hat{y}_i$. We use some rule to define if the prediction is a success or a failure.
The number of successes X then follow a binomial distribution with M trials and success probability p, and

$$\hat{p} = \frac{X}{M}$$

with mean p and variance $\frac{p(1-p)}{M}$.

A common way to construct a confidence interval for the success probability is to use the normal approximation

$$Z = \frac{\hat{p} - p}{\sqrt{\frac{\hat{p}(1-\hat{p})}{M}}} \sim N(0,1)$$

which gives the $(1-\alpha)100‰$ confidence interval

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{M}}$$

Exact versions (not using normality) are the Clopper-Pearson or Agresti-Coull intervals.

```
## [1] "lasso"
```

```
## [1] "Normal approx CI"
```

```
## [1] 0.7420393 0.8302498
```

```
## [1] "Clopper Pearson CI"
```

```
##
##  Exact binomial test
##
## data:  X and M
## number of successes = 261, number of trials = 332, p-value < 2.2e-16
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.7380713 0.8290302
## sample estimates:
## probability of success
##              0.7861446
```

```
## [1] "randomforest"
```

```
## [1] "Normal approx CI"
```

```
## [1] 0.7291144 0.8190783
```

```
## [1] "Clopper Pearson CI"
```

```
##
##  Exact binomial test
##
## data:  X and M
## number of successes = 257, number of trials = 332, p-value < 2.2e-16
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.7252650 0.8179625
## sample estimates:
## probability of success
##              0.7740964
```

McNemar´s test

Is method A different from method B?

Consider M pairs of observations (predictions from method A and B) in the test set, and classify as

- successes (1) - for correct classifications
- failures (0) - for wrong classifications

(The true response y in the test set is used to define success and failure.)

The pairs are assumed to be independent, but the two observations within a pair may be dependent. We call this *matched pairs*.

The numbers $(X_{01}, X_{10}, X_{00}, X_{11})$ of the four possible outcomes of each pair, 01, 10, 00 and 11, respectively, are assumed to follow a multinomial distribution with parameters $(N; q_{01}, q_{10}, q_{00}, q_{11})$.

To test the null hypothesis that the probability of success in the first observation of a pair is equal to the probability of success in the second observation (the two methods have the same performance)

$$q_{10} + q_{11} = q_{01} + q_{11}, \text{ or } q_{10} = q_{01}$$

McNemar's test statistic,

$$T(X_{01}, X_{10}) = (X_{01} - X_{10})^2/(X_{01} + X_{10})$$

is often used, with large values indicating rejection (McNemar, 1947, Agresti, 2002 pp. 410–412).

Asymptotically T follows a X^2 distribution with 1 degree of freedom when the null hypothesis is true.

The sum $X_{01} - X_{10}$ need to be large (rule of thumb at least 25), unless a two-sided binomial version of the test is recommended (with $n = X_{01} - X_{10}$ and $p = 0.5$ and number of successes equal X_{01}).

An exact conditional p-value can also be calculated by enumeration.

```
tab=table(classlasso==test$diabetes,classrf==test$diabetes)
tab
```

```
##
##          FALSE TRUE
##   FALSE    51   20
##   TRUE     24  237
```

```
mcnemar.test(tab,correct=FALSE)
```

```
##
##  McNemar's Chi-squared test
##
## data:  tab
## McNemar's chi-squared = 0.36364, df = 1, p-value = 0.5465
```

```
binom.test(tab[1,2],n=tab[1,2]+tab[2,1],p=0.5)
```

```
##
##  Exact binomial test
##
## data:  tab[1, 2] and tab[1, 2] + tab[2, 1]
## number of successes = 20, number of trials = 44, p-value = 0.6516
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.3039071 0.6115279
## sample estimates:
## probability of success
##              0.4545455
```

Conclusion: with respect to using 0.5 as cut-off for classification as disease then the paired McNemar two-sided test that lasso and RF produce equally good results is not rejected at level 0.05.

If we have more than two methods to compare, the Cochrane Q-test can be used. Wikipedia

Confidence intervals for paired proportions

Confidence interval for the difference between success-proportions can be calculated using for example an asymptotic Wald interval. See Fagerland et al (2014) for this and other choices, not R package but see references for R-scripts.

The package `ExactCIdiff` is explained in the R Journal

ROC-AUC

See L3 for definitions of sensitivity and specificity.

The receiver operating characteristics (ROC) curve gives a graphical display of the sensitivity against specificity, as the threshold value (cut-off on probability of success or disease) is chosen over the range of all possible values. An ideal classifier will give a ROC curve which hugs the top left corner, while a straight line represents a classifier with a random guess of the outcome.

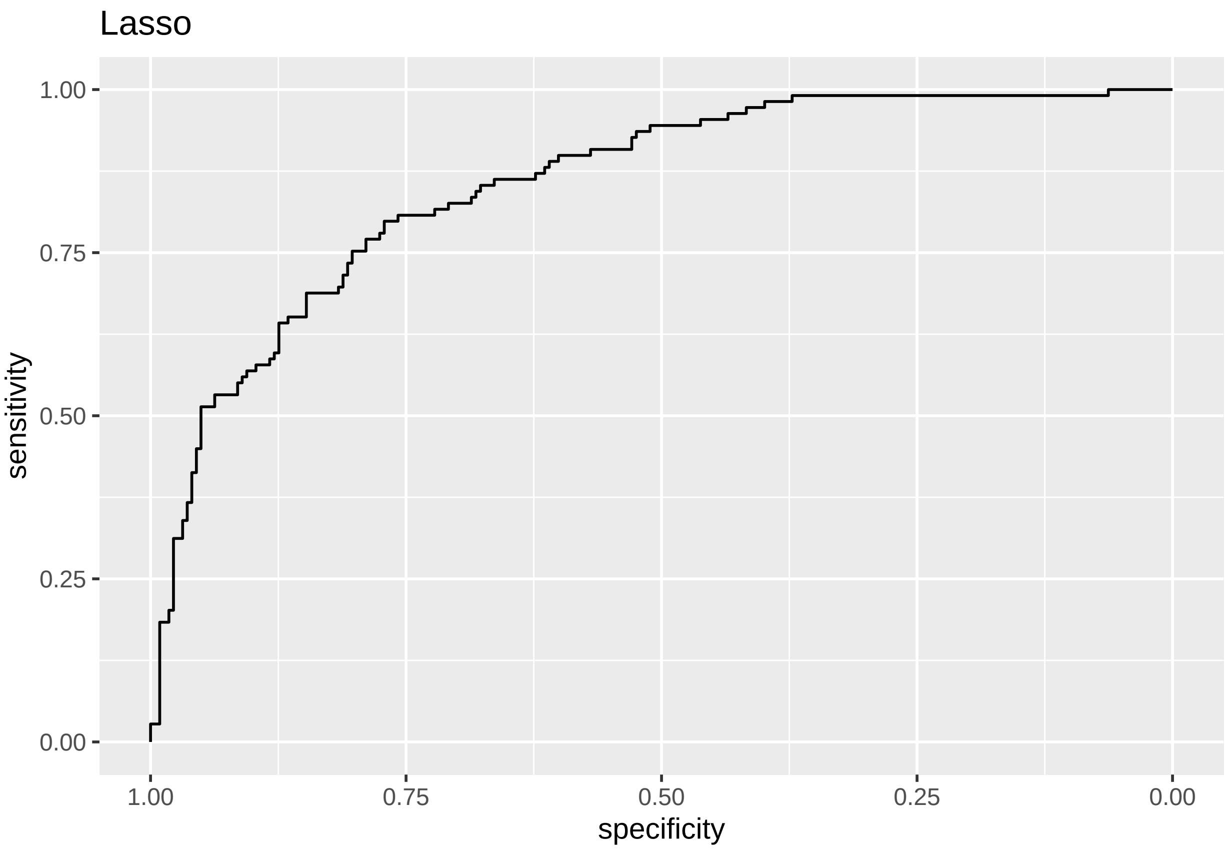
- The **ROC-AUC** score is the area under the ROC curve - calculated using the trapezoid rule. It ranges between the values 0 and 1, where a higher value indicates a better classifier.
- The AUC score is useful for comparing the performance of different classifiers, as all possible threshold values are taken into account.
- If the prevalence (case proportion) is very low (0.01ish), the ROC-AUC may be misleading, and the PR-AUC is more commonly used.

The ROC-AUC (based on the trapezoid rule) can be seen to be equal to the nonparametric Wilcoxon-Mann-Whitney statistic (DeLong et al 1988).

In the R package `pROC` several methods to produce confidence intervals for the ROC and ROC-AUC exists, and tests to compare ROC-AUC from two methods (paired or unpaired).

Below we use:

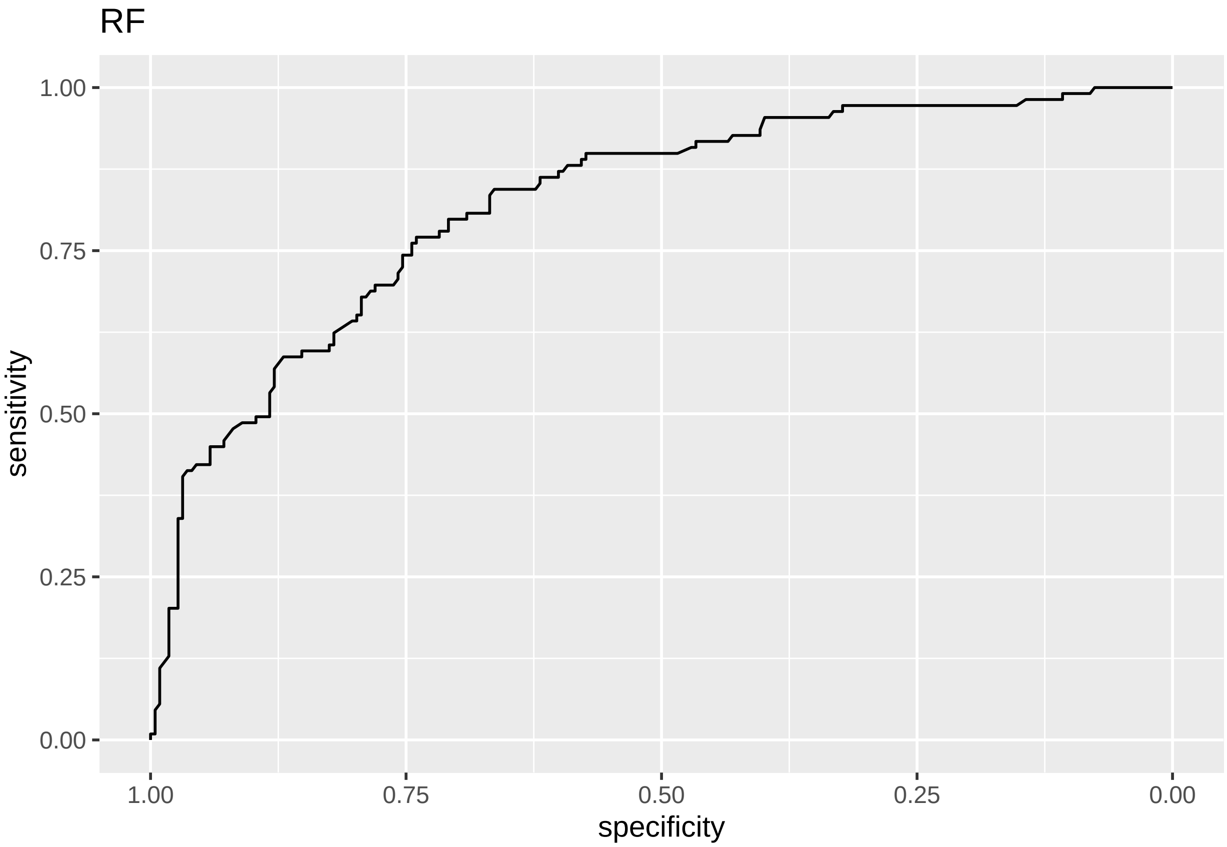
- DeLong et al confidence intervals for the ROC and the ROC-AUC for each prediction method.
- DeLong et al test for two paired (correlated) ROC curves. This test is based on asymptotic normal theory for the U-statistic.



```
## [1] "Lasso ROC-AUC with CI"
```

```
## Area under the curve: 0.8544
```

```
## 95% CI: 0.8125-0.8964 (DeLong)
```



```
## [1] "RF ROC-AUC with CI"
```

```
## 95% CI: 0.7746-0.8698 (DeLong)
```

```
## [1] "Comparing AUC for lasso and RF"
```

```
##
##  DeLong's test for two correlated ROC curves
##
## data:  lassoroc and rfroc
## Z = 2.2423, p-value = 0.02494
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.8544452   0.8221911
```

Conclusion: with respect to ROC-AUC then the two-sided test that lasso and RF produce equally good results is rejected at level 0.05.

Observe that the RF is significantly better than lasso wrt ROC-AUC, but not wrt misclassifiation error.

Regression

For regression we would like to focus on providing an estimate for the Err_T for a squared error rate.

$$\mathsf{Err}_T = \mathsf{E}[L(Y, \hat{f}(X)) \mid T]$$

Here the expected value is with respect to (X, Y), but the training set is fixed - so that this is the test set error is for this specific training set T.
In ELS Ch7.1 we saw that the *mean squared error on the test set* was a natural estimator.

In the unconditional version, we take expected value over ALL that is random - including the training set

$$\mathsf{Err} = \mathsf{E}(\mathsf{E}[L(Y, \hat{f}(X)) \mid T]) = \mathsf{E}_T[\mathsf{Err}_T]$$

However, we did not work to provide an estimate of the *variability* of this estimate - or how to provide a confidence interval for Err_T.

Let the mean squared error on the test set be denoted $\widehat{\text{MSEP}}$.

If we can assume that the “residuals” on the test set $y_i - \hat{y}_i$ follow a normal distribution with some mean μ_i and some variance σ_i^2, then there is a relationship between the$\widehat{\text{MSEP}}$ and a sum of non-central χ^2 distributions, see Faber (1999). However, it is not clear how to turn that into a confidence interval for Err_T.

Not seen in literature: Another possibility is to use bootstrapping on the “test set residuals”. This can provide a bootstrap confidence interval for the Err_T. With bootstrapping it would also be possible to look at randomly flipping the A and B method to get the distribution of the $\widehat{\text{MSEP}}$ under the null hypothesis that the two methods are equal, and use tha percentage of times the bootstrap samples are larger than the observed $\widehat{\text{MSEP}}$ to be the p-value.

I have really not found relevant literature - and I am afraid that I may have missed out on good solutions here. Please contact me if you know of good solutions to this problem.

Data poor (small sample) situation

We may also refer to this as a small sample situation, and in this case we need to resort to resampling to get an estimate of the mean squared error , misclassification error, or similar - on “new” data.

We are again interested in estimating the conditional (on the training data) Err_{T}, but as we saw in ELS Ch 7.10-7.11, using resampling techniques we will instead be providing an estimate for the unconditional Err.

That might of cause be ok for us.

We have in ELS Ch 7 looked at cross-validation and bootstrapping.

Cross-validation

Remember from ELS Ch 7.10 that with cross-validation the Err estimate:

- The allocation of observation $\{1, \dots, N\}$ to folds $\{1, \dots, K\}$ is done using an indexing function $\kappa : \{1, \dots, N\} \rightarrow \{1, \dots, K\}$, that for each observation allocate the observation to one of K folds.
- Further, $\hat{f}^{-k}(x)$ is the fitted function, computed on the observations except the kth fold (the observations from the kth fold is removed).
- The CV estimate of the expected prediction error $\text{Err} = \text{E}_T \text{E}_{X^0, Y^0}[L(Y^0, \hat{f}(X^0)) \mid T]$ is then

$$\text{CV}(\hat{f}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i))$$

Can the validation fold results be handled like the test set?

Question:

Can we handle the predictions in the hold-out folds $\hat{y}_i$ as *independent predictions* at the observations x_i - as we did in the data rich situation above (when we had a separate test set and used the “same” full training set for fitting the model)?

To address this a simulation study is conducted. Here

- data are simulated to follow a simple linear regression.
- $N = 50$.
- The observations are divided into 5 fold of 10 observations.
- Then a 5-fold CV is performed where a simple linear regression is fitted on the training folds and predictions are performed in the test fold.
- Residuals are then formed for the test fold.

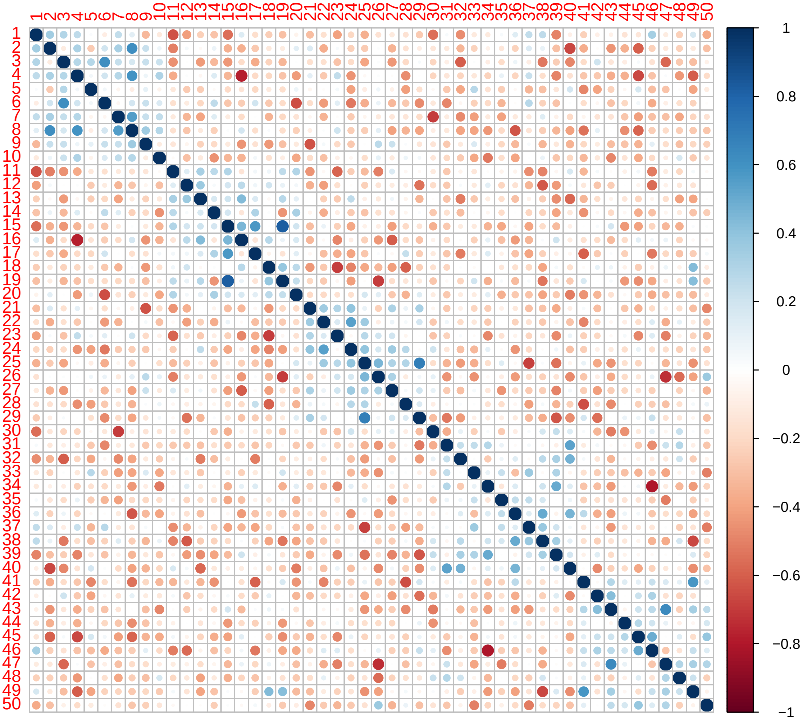
The simulations are repeated B=1000 times, and correlation between the N residuals for the test folds are calculated.

The question to be checked is if the residuals for observations in the same fold are correlated in a different way than residuals in different folds. If that is the case, then the residuals can not be seen to be independent, and standard methods to construct CI and perform a test is not valid.

The correlation plot shows 6*correlation (just to get colours stronger) for the residuals (difference between prediction and truth) between observations within and between folds.

There are 10 observations in each of 5 folds - ordered so that observations labelled 1-10 is fold 1, observation 11-20 is fold 2 etc. Observe that the correlation matrix has a block diagonal structure where the trend is that the observations in the same fold are slightly (numbers divided by 6) positive correlated, while the observations from different folds (away from the diagonal) are slightly negatively correlated (again divide the numbers by 6).

This means that the residuals from a 5-fold CV can not be seen to be independent across all observations, but will exhibit slight positive and negative correlations.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

```
## [1] "Mean correlation within folds"

## [1] 0.02533422

## [1] "Mean correlation between folds"

## [1] -0.03049688

## [1] 12
```

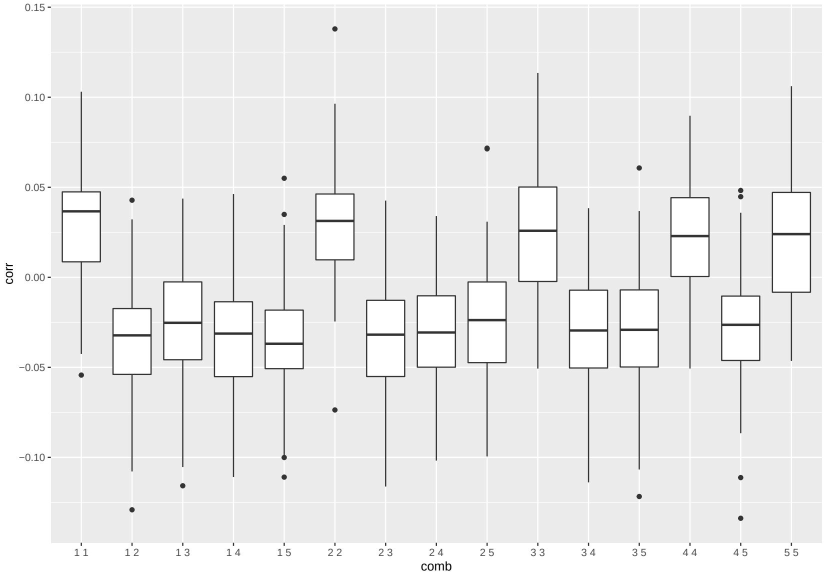
The boxplot of the correlation between residuals are taken between two folds, labelled on the horizonal axes.

There are $50 * 49/2 = 1225$ unique pairs of observations (residuals) for the simulated example. There are 5 folds and the average correlation for the 5 times $10 * 9/2 = 45$ pairs = 225 pairs within each fold is 0.0253342.

The average correlation for the 1000 pairs between folds is -0.0304969.

However - testing if the correlation is different from null for all possible pairs of observation of the residuals (with 50 observation we have $50 * 49/2$ pairs), only gave a significant result for 12 using FDR cut-off 0.05.

Most articles state that this is a substantial problem, mainly because for constructing tests the variance of the test statistics is underestimated with positively correlated tests. However, other articles like Wong and Yang (2017) do not consider this a problem.


0.15
0.10
0.05
0.00
−0.05
−0.10
corr
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
5 5
comb

What can we present from the CV?

We have now focus on some loss function, like squared loss, binomial deviance, cross-entropy loss.

When we performed model selection with CV for the lasso we plotted some mean and standard error. How did we then calculate the standard error and the mean? Can we use this standard error to calculate a confidence interval?

We had N observations in the training set and choose K-fold CV:

$$\mathsf{CV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-k(i)}(x_i))$$

Assuming that $N = K \cdot N_K$ so that the number of observations in each fold N_j is the same and equal to N_K.

$$\mathsf{CV}(\hat{f}) = \frac{1}{K}\sum_{j=1}^{K}\frac{1}{N_K}\sum_{i\in k(i)} L(y_i, \hat{f}^{-k(i)}(x_i)) = \frac{1}{K}\sum_{j=1}^{K}\widehat{\mathsf{CV}}_j$$

What we plotted was the $\frac{1}{K}\sum_{j=1}^{K}\widehat{\mathsf{CV}}_j$ as the estimator for the evaluation criterion, and then ± 1 standard error of this mean.

The variance of the mean was estimated as

$$\mathsf{SE}^2(\hat{f}) = \frac{1}{K}(\frac{1}{K-1}\sum_{j=1}^{K}(\widehat{\mathsf{CV}}_j - \mathsf{CV}(\hat{f}))^2)$$

Since the residuals within a fold are positively correlated and between folds are negatively correlated, we only present plots of

$$\mathsf{CV}(\hat{f}) \pm \mathsf{SE}(\hat{f})$$

and are happy with that.

ROC-AUC on CV data

For the ROC-AUC two different strategies are possible:

- For each CV fold separately calculate the ROC-AUC, and then report average and standard error (as above) over the fold. This is called *average approach*.
- Use all predictions (across all folds) to calculate ROC_AUC. This is called *pooled approach*. Then results from the DeLongi method might not be completely correct due to the observations being positively correlated within folds and negatively correlated between folds.

Airola et al (2010) suggest an hybrid combination of the two methods.

5x2 cross-validation

Dietterich (1998) might have been one of the first to point out the problems with the non-independence between observations from different folds.

Strategy:

- First divide the data set into two equally sized sets. Use one as training set and one as validation set, and then swap the role of the two.
- There is no overlap between these sets, so the idea is that the two sets of predictons on validations set are independent (but again, different estimated models are used on each part).
- The reshuffle all data, and do the same again. Now you have results from “4 folds”.
- Repeat three more times - and now you have “10 folds” (or, 5 times 2-fold CV used together).

The choice of 5 repetitions of 2-fold cross-validation is according to Dietterich (1998) that the overlap between the 5 repetions is not very large, but adding more repetitions will again give “too dependent” data. For fewer than 5 repetitions it will be hard to construct tests from these data (and constructing a test is the aim of Dietterich).

5x2 paired t-test

Dietterich (1998) only used the 5x2 CV set-up for comparing two classification prediction methods A and B.
Then the test is based on a paired t-test on the difference in error rates of the two classifiers on each fold. A lot of work has done into trying to get the most correct variance for this test. For formula and details see Dietterich (1998) or Alpaydin (2014) Ch 19.11.3.

Other solutions

Comparing misclassification rates for two prediction models: Dietterich (1998) show in a simulation study that using McNemars test on the *training data* gives a valid test (in the situations he studied).