MA8701 Advanced methods in statistical inference and learning

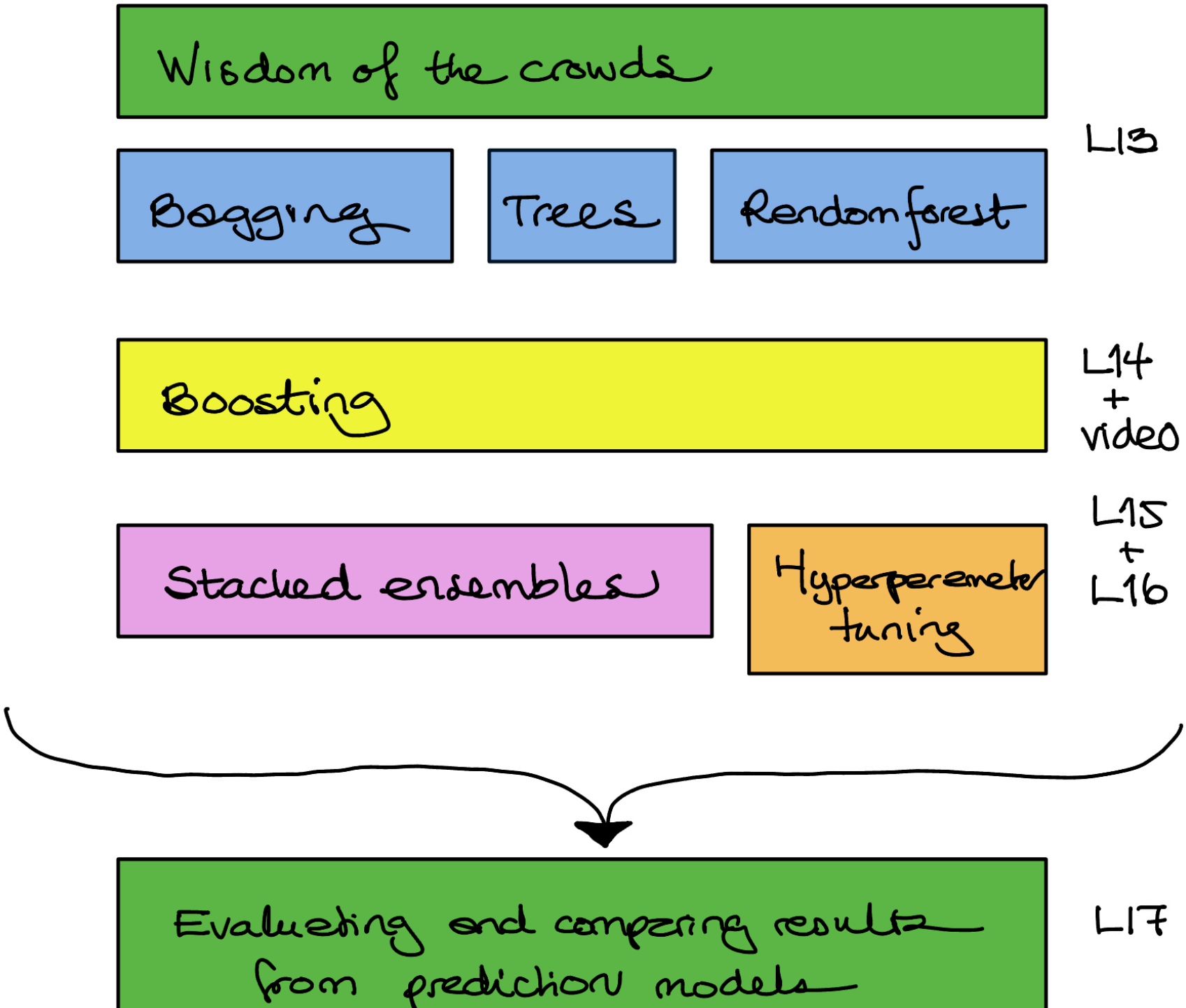
Part 3: Ensembles. L16: Hyperparameter tuning

Mette Langaas

Lectured 10.03.2023

~~3/8/23~~

Before we start



Before we start

Literature

- Hyperparameter tuning with Bayesian Optimization. Frazier (2018): "A tutorial on Bayesian optimization", https://arxiv.org/abs/1807.02811: Sections 1,2,3,4.1, 5: only the section "Noisy evaluations", 6,7.
- 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.

Choosing hyperparameters : Group discussion

Parameters that can't be estimated DIRECTLY from data.
→ affect model fit
→ decided by the user

- What are *hyperparameters*?
- Which hyperparameters have we encountered in the course so far?
- What are challenges with hyperparameter tuning?

CV: k-fold
k-NN: k
elastic net: α, λ
trees: depth, split crit Gini/mse/..
RF: # tree

boosting: learning rate
max depth
subsample row col
B = # tree

discrete
continuous
categorical

Challenges:

- expensive & computationally ↙ many hyperpar d
  ↑ fit our model
- need a separate validation set or CV
- hyperparam - may direcly relate to bias-variance tradeoff
  ||
  penalization, shrinkage
- optimization problem ÷ we optimize some loss
  misclass
  ↓ MSE
  ↙ ROC-AUC

  not able to calc gradients and Hessians
  ||
  black box
- unclear which of many possible hyperpar to tune

There exist many ways to *group* methods for hyperparameter tuning. One way to look at this is (Kuhn and Silge, 2021, Ch 12)

- grid search: specify a set of possible values a priori and investigate only these values, choose the value where the chosen selection criterion is optimal. This is also called "model free methods".
- iterative search: start with a set of values, fit/evaluate some (surrogate) model (might also be the loss function), and based on this choose new values to evaluate next.

For grid search also methods for *speeding up calculations* exists - for example by stopping evaluation at a grid point where the loss is seen to be high after some CV-folds, for example the method of *racing* described by Kuhn and Silge, Ch 13.4.

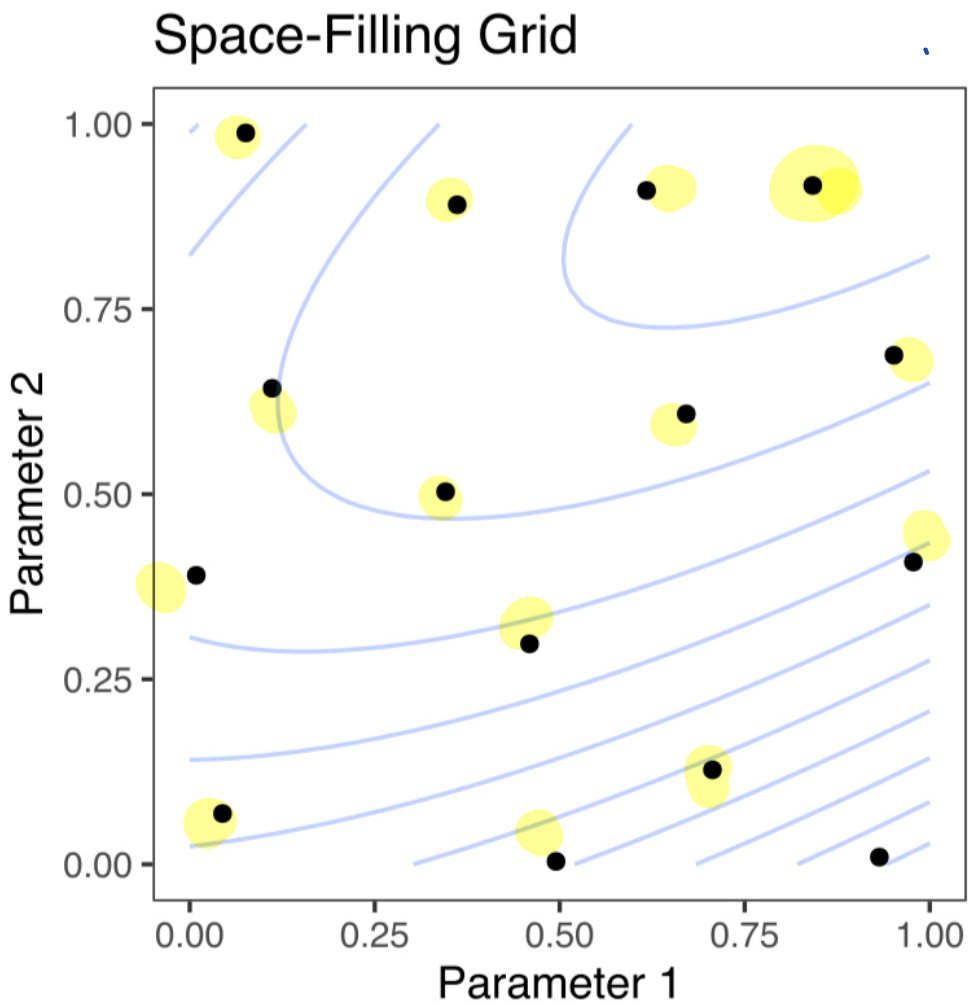
- Some AutoML performs hyperparameter tuning "in the hood" → often grid search
- Stacked ensembles: make ensemble from many diff. hyperpar. models

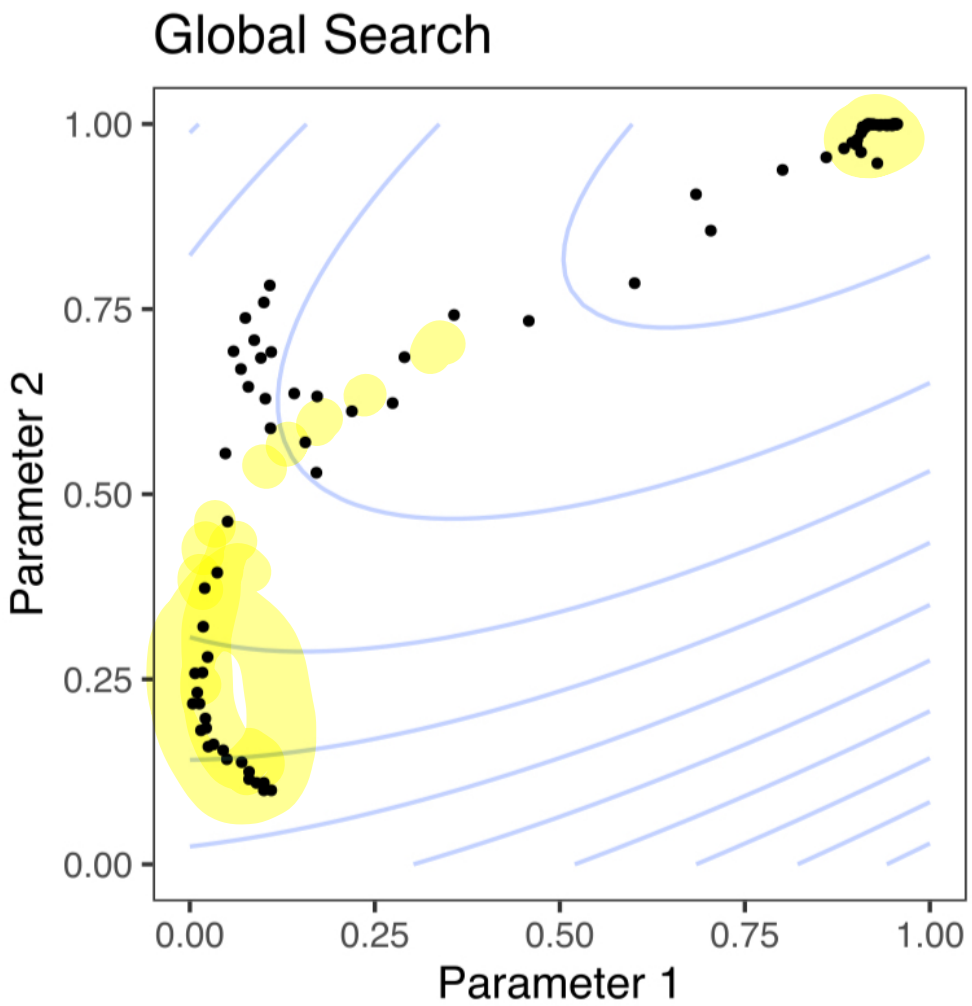
Kuhn & Silge (2021)

Grid search vs iterative search

GRID SEARCH

ITERATIVE SEARCH





Decide first on which values of the hyperpar. to investigate → choose the best combination

Start with a small grid. Fit some surrogate model - use that to suggest new point(s)

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.

Bayesian optimization (BO)

Criterion to maximize

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

X : continuous hyperparameters $x \in \mathbb{R}^d$ $d \leq 20$

typically is a hyperrectangle

$f(x)$: objective function to maximize from some CV model fit at x

$f(x)$: expensive

lack specific structure (linear, convex)

not possible to do $f'(x)$, $f''(x)$

AIM: global - not local optimum.

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 our problem.

We now look at the multivariate Gaussian distribution and conditional distribution, a Gaussian process

Gaussian process

n = # combs of hyperpar

1) Observe $x_1, x_2, \ldots, x_n$ values in hyperp. space

$f(x_1)$ $f(x_2)$ $f(x_n)$ ← values of obj function

2) Model $f(x)$

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim N_n(\underset{n\times 1}{\mu}, \underset{n\times n}{\Sigma})$$

Σ ← dependent on distance between the x's

Y's

Gaussian processes

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

A Gaussian process is defined for

- times or locations x_i, $i = 1, \ldots, n$ in $\Re^d$, where
- $Y_i = Y(x_i)$ is a random variable at x_i ← objective function
- such that $Y = (Y_1, \ldots, Y_n)$ 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 $\mathrm{Var}(Y(x)) = \sigma^2$ 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.

assume this

Correlation functions

(Eidsvik 2017, page 7, Frazier 2018, Ch 3.1)

Correlation functions are also referred to as *kernels*.

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

Power exponential or Gaussian kernel

$$\text{Corr}(Y(x), Y(x')) = \exp(-\phi_G \|x - x'\|_2^2)$$

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

exponential $\text{Corr}(Y(x), Y(x')) = \exp(-\phi_E |x - x'|)$

Matern-type kernel

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

now with decay-describing parameter ϕ_M.

The parameters of the kernels need to be estimated, see Ch 3.2 of Frazier 2018 (who use a slightly different parameterization). We will just assume that these parameters are known.

(Class notes: study Figure 4 and 5 of Eidsvik, 2018.)

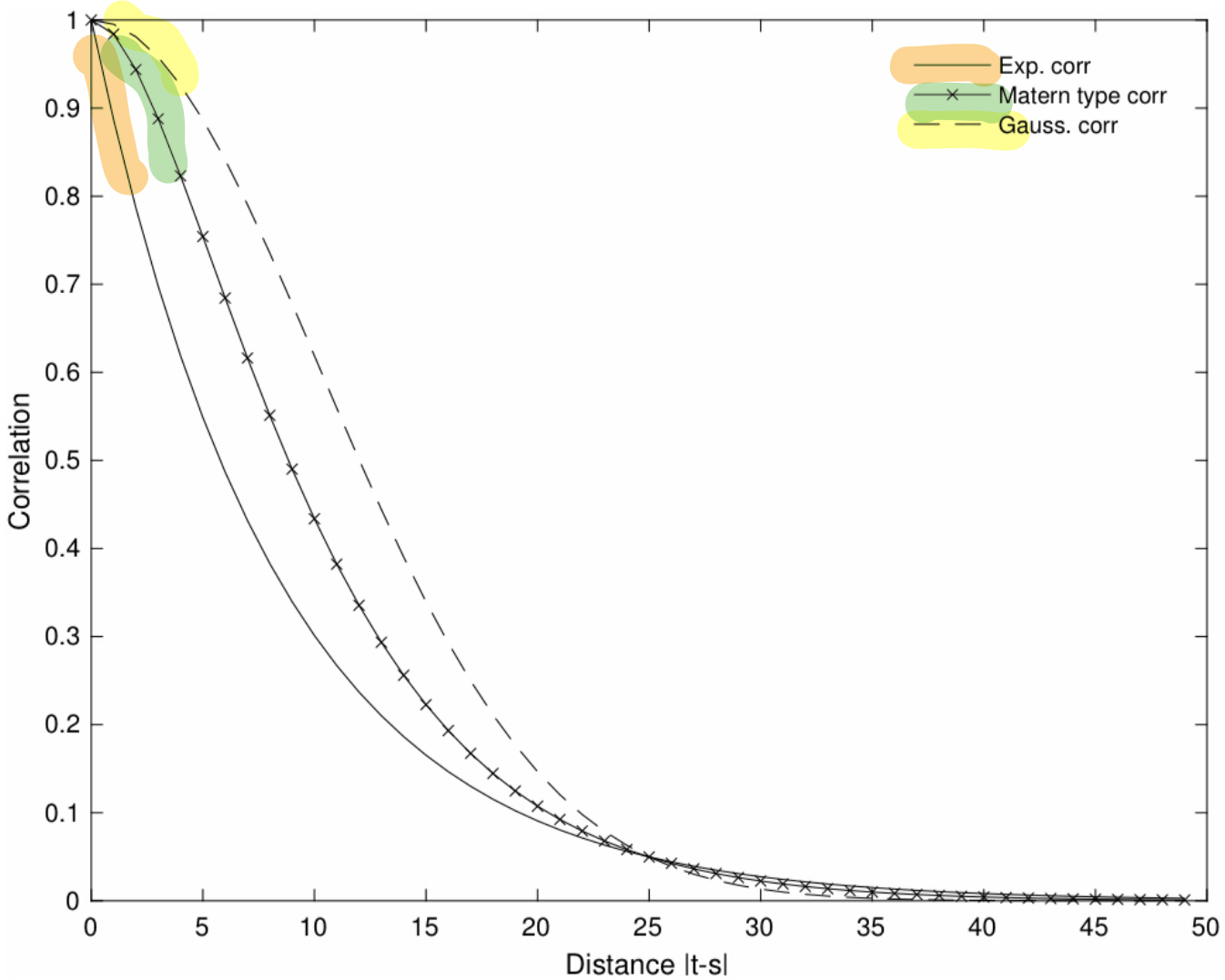


Figure 4: Three different correlation functions. Gdsun (2018)

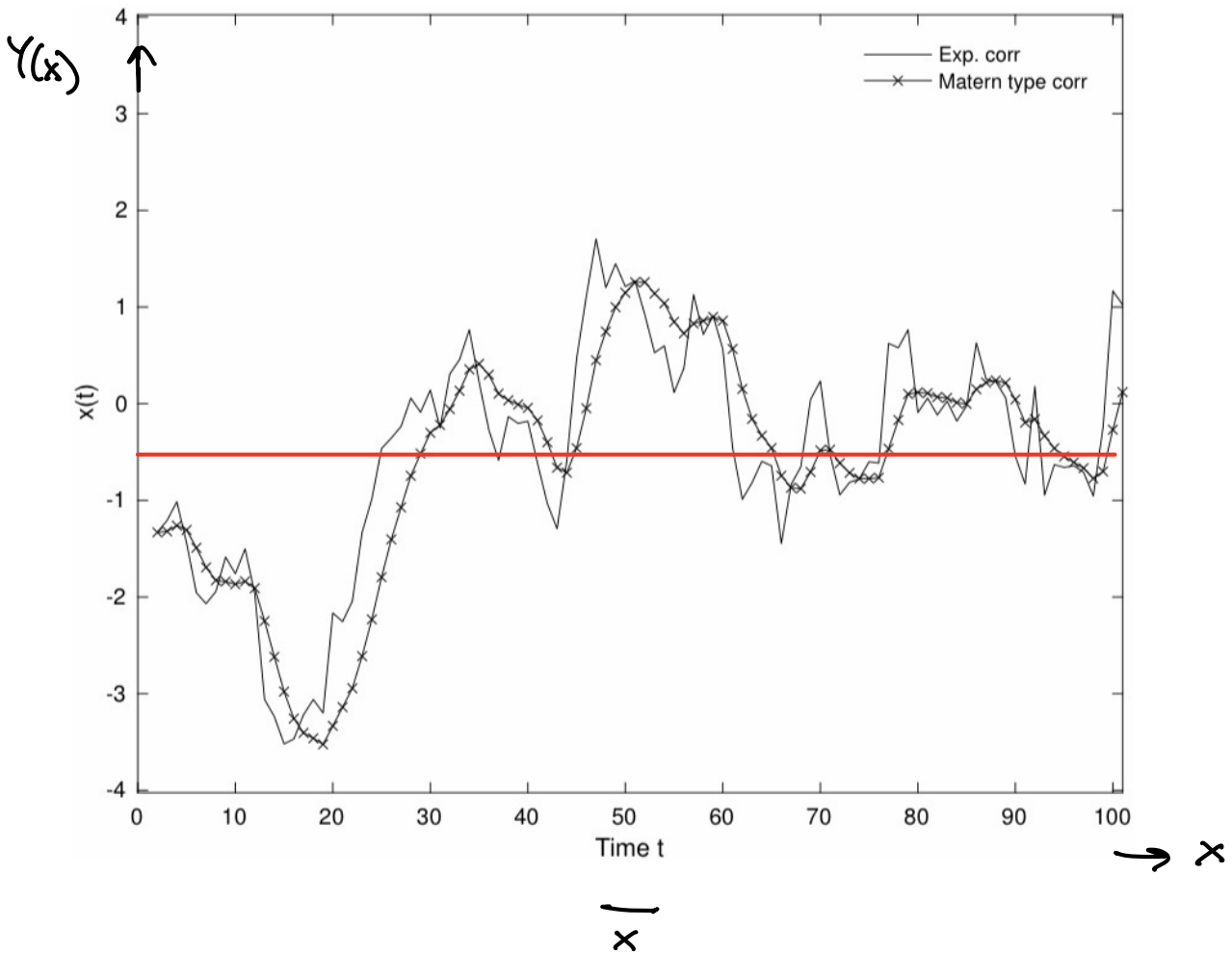


Figure 5: One realization from the Gaussian process with exponential covariance function and one with Matern type correlation function. The mean is 0 and variance 1. The correlation decay parameters are $\phi_E = 3/25$ and $\phi_M = 0.19$.

Edwin (2017)

From correlations into covariance matrix

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

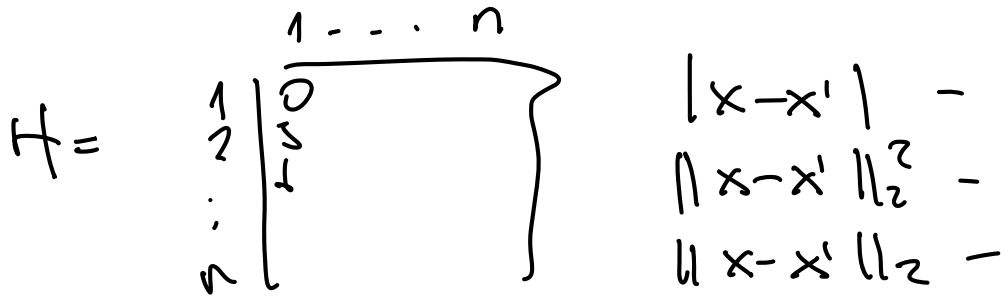
To get from correlation function to a $n \times n$ covariance matrix first construct a $n \times n$ matrix of distances for each pair of positions, denote this H.

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

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

where $\otimes$ is elementwise multiplication.

See Eidsvik (2018, Ch 3.2 and 3.3) for how to build covariance matrices in an efficient way.



So far: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where $y_i = f(x_i)$

$x_i \in \mathbb{R}^d$

y_i: objective function based on CV

Now now

$$Y \sim N_n(\mu, \Sigma)$$

Y: $f(x)$'s, $n \times 1$

Σ: correlation function combined with k-distances

μ: most common to set $\mu(x) = \mu$

$\eta = (\mu, \phi_M, \sigma^2)$

ϕ_E

ϕ_G ← chosen or estimated

decided

Focus now: where to sample next?

x

$Y = Y_1$ and a new point is Y_2

Y: $n \times 1$

Y_2: 1×1

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:

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

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

$$\mathbf{Y}_2 \mid (\mathbf{Y}_1 = \mathbf{Y}_1) \sim N_{p2}(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{Y}_1 - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}).$$

objective function at new point x_2 in $\mathbb{R}^d$

$E(Y_2|Y_1)$ $\mu_n(x)$

$Cov(Y_2|Y_1)$ $\sigma_n^2(x)$

95% Bayesian credible: $\mu_n(x) \pm 1.96\,\sigma_n(x)$

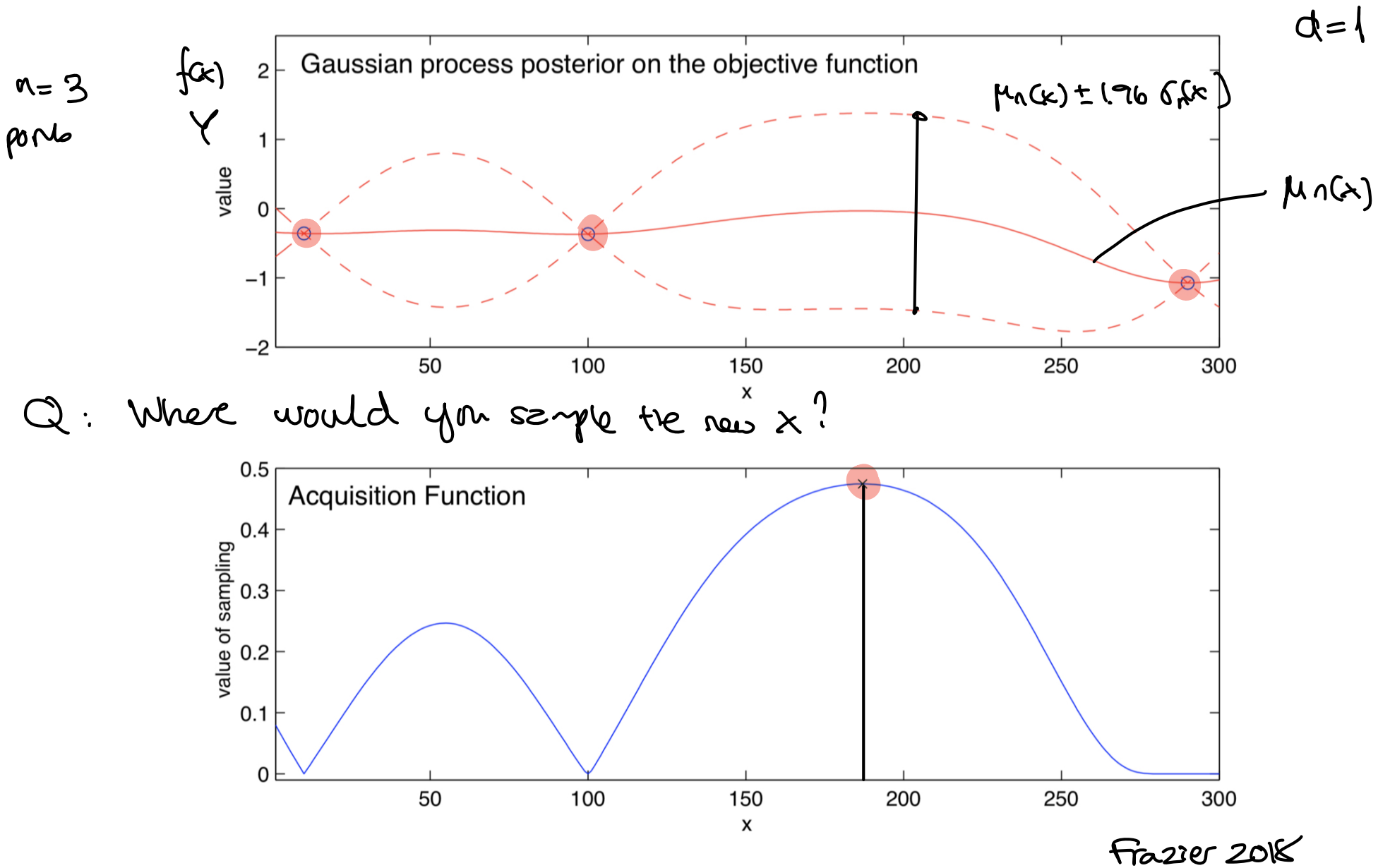


Figure 1: Illustration of BayesOpt, maximizing an objective function f with a 1-dimensional continuous input. The top panel shows: noise-free observations of the objective function f at 3 points, in blue; an estimate of $f(x)$ (solid red line); and Bayesian credible intervals (similar to confidence intervals) 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."

Acquisition function: Expected improvement

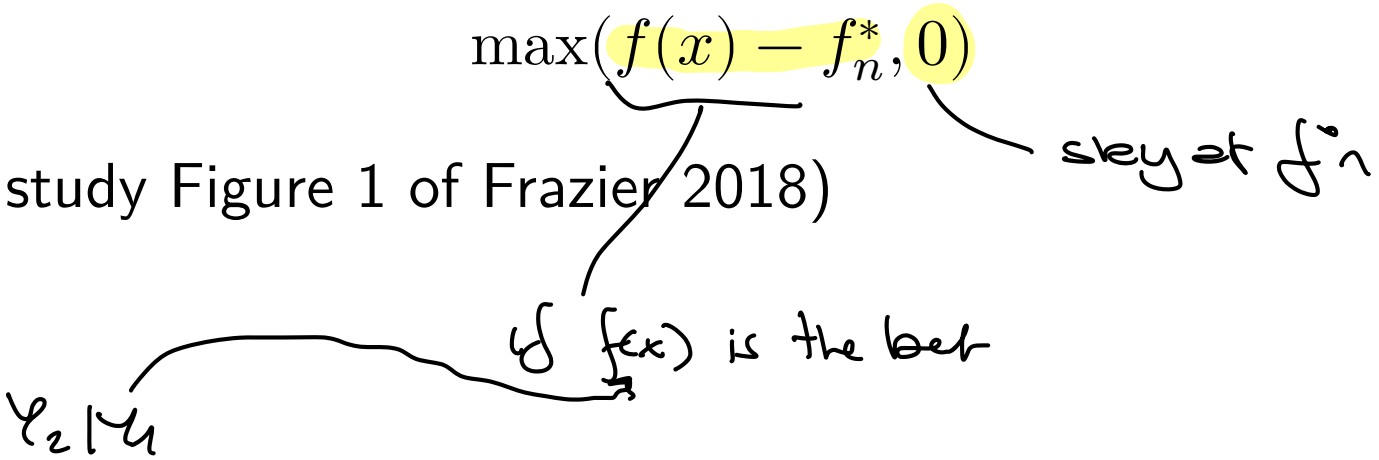
(Frazier 2018 page 7)

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

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

(In class study Figure 1 of Frazier 2018)



3) We define the *expected improvement* as

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

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 $\sigma_n^2(x)$.

$$\int_{-\infty}^{\infty} \max(\tilde{f}(x) - f_n^*, 0) \cdot \underbrace{f(y_k \mid y_1)}_{N(\mu_n(x), \sigma_n^2(x))} \, dy_k$$

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

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

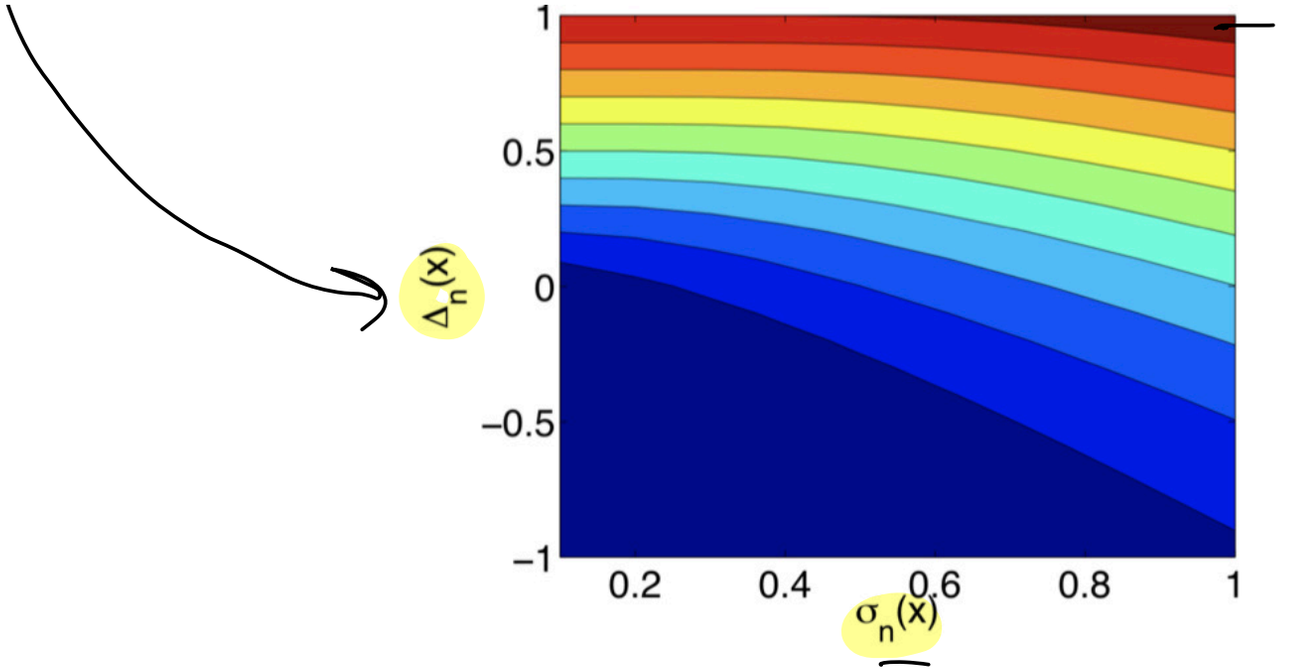
$$-\mathrm{abs}(\mu_n(x) - f_n^*)\Phi\left(\frac{\mu_n(x) - f_n^*}{\sigma_n(x)}\right)$$

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

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

$$x_{n+1} = \mathrm{argmax}\mathrm{EI}_n(x)$$

Is often found using quasi-Newton optimization.



Frazier 2018

Figure 3: Contour plot of EI(x), the expected improvement (8), in terms of $\Delta_n(x)$ (the expected difference in quality between the proposed point and the best previously evaluated point) and the posterior standard deviation $\sigma_n(x)$. Blue indicates smaller values and red higher ones. The expected improvement is increasing in both quantities, and curves of $\Delta_n(x)$ versus $\sigma_n(x)$ with equal EI define an implicit tradeoff between evaluating at points with high expected quality (high $\Delta_n(x)$ versus high uncertainty (high $\sigma_n(x)$).

trade off between high expected performance and high uncertainty

exploitation

exploration

Algorithm for Bayesian optimization of a function f

(Frazier 2018, page 3, noise-free evaluation)

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

Explain: what does these step mean !

What does the steps mean?

- Gaussian prior: choose (estimate?) mean and correlation function for the problem.
- Observe n_0 points: calculate the loss function at each of the points (remark: we have noise)
- Update the posterior: calculate the conditional distribution for f for a new point given the observed loss at all previously observed points
- Acquisition function: find argmax$\mathsf{EI}_n(x)$.

(Class notes: Figure 1 of Frazier 2018.)

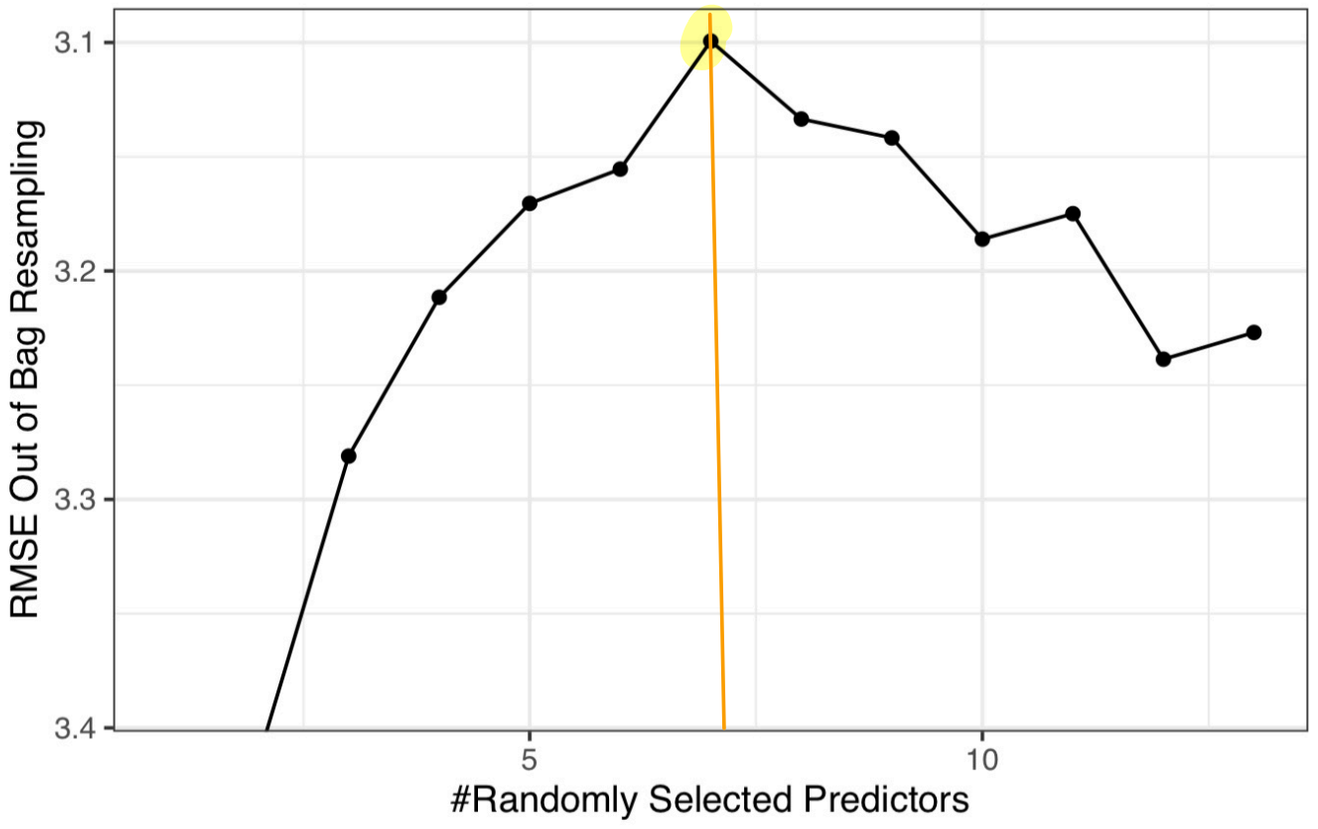
- For a point x we model the distribution of $f(x)$,
- which is normally distributed with mean $\mu_n(x)$ and variance $\sigma_n^2(x)$. The mean and variance is found from the conditional distribution.
- With 95% credibility interval $\mu_n(x) \pm 1.95\sigma_n(x)$.
- The width of the credibility interval at observations is 0.

2.3.1 Example

(Kuhn and Silge, Ch 14, the example is for SVM)

First just grid search to test what is best value for `mtry`

```
data(Boston, package = "MASS")
# first using a grid
tune_grid <- expand.grid(
  mtry = (1:13))
#  ntree=seq(100,500,length=10)) # how to also include ntree? primary only mtry, how to de
tune_control <- caret::trainControl(
  method = "oob", # cross-validation #eller cv
  #number = 3, # with n folds
  verboseIter = FALSE, # no training log
  allowParallel = FALSE # FALSE for reproducible results
)
rf_tune <- caret::train(
  medv~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+black+lstat,
  data=Boston,
  na.action=na.roughfix,
  trControl = tune_control,
  tuneGrid = tune_grid,
  method = "rf", # rf is randomForest, checked at #vhttp://topepo.github.io/caret/train-mo
  verbose = TRUE
)
tuneplot <- function(x, probs = .90) {
  ggplot(x) +
    coord_cartesian(ylim = c(quantile(x$results$RMSE, probs = probs), min(x$results$RMSE))
    theme_bw()
}
tuneplot(rf_tune)
```

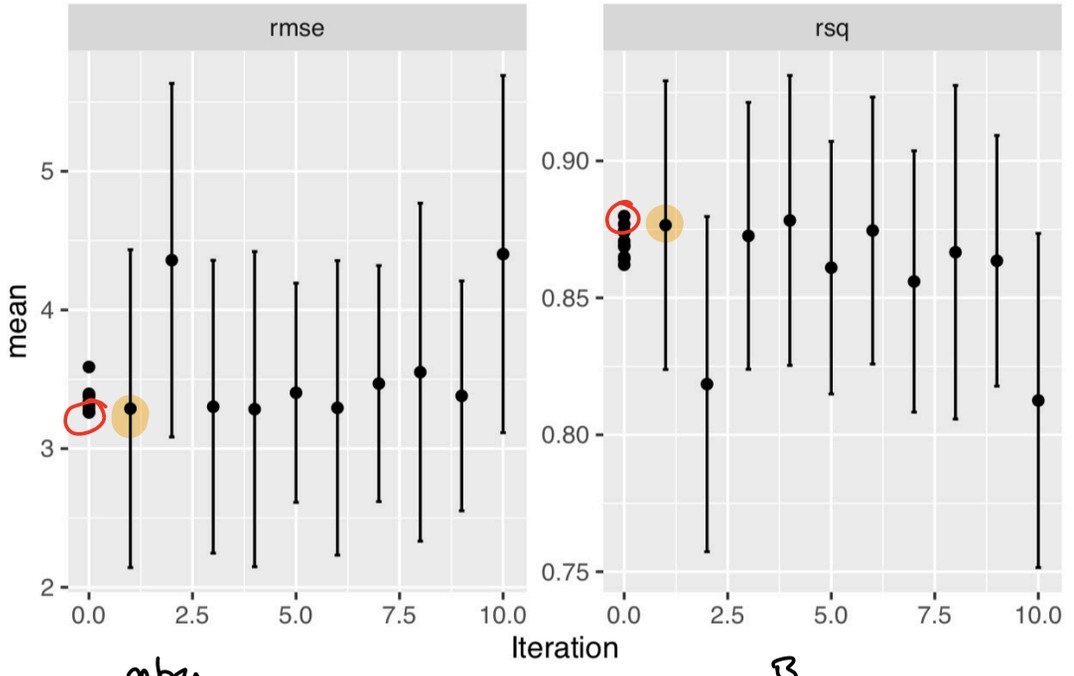



```
rf_tune$bestTune
```

```
  mtry
7    7
```

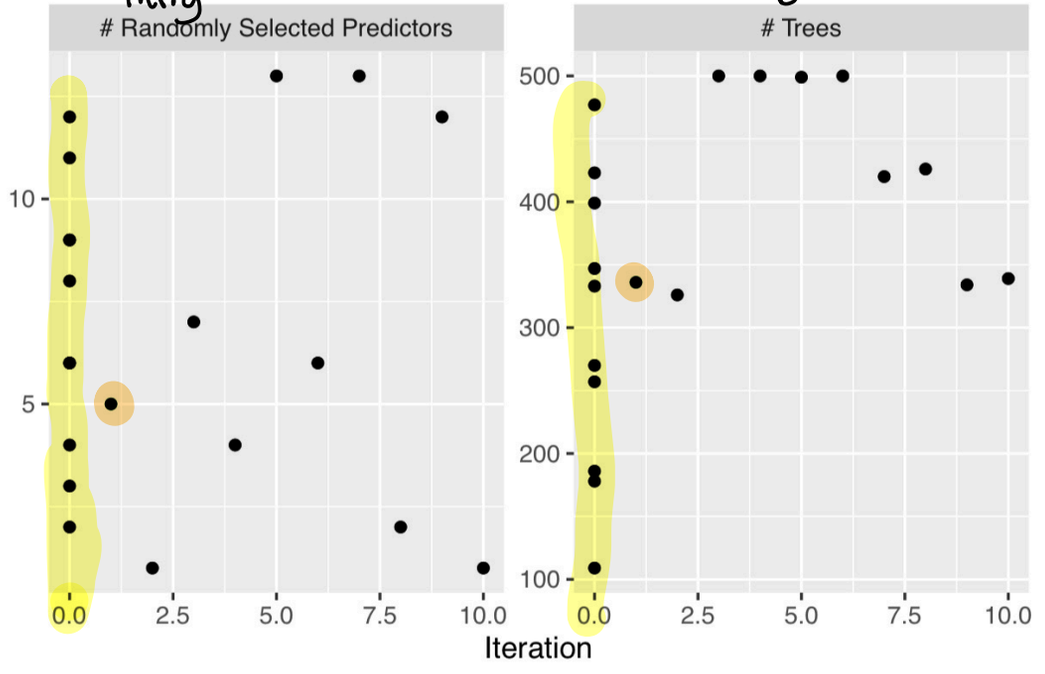
The R the function `tune_bayes` is available in the package `tune`, and requires that the analyses is done with a workflow. Default in the GP is exponential correlation function, but first we try the Matern.

```
tree_rec <- recipe(medv~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+black+lstat, d

tune_spec <- rand_forest( # parsnip interface to random forests models
  mode="regression",
  mtry = tune(),
  trees = tune(),
#  min_n = tune()
  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")
```

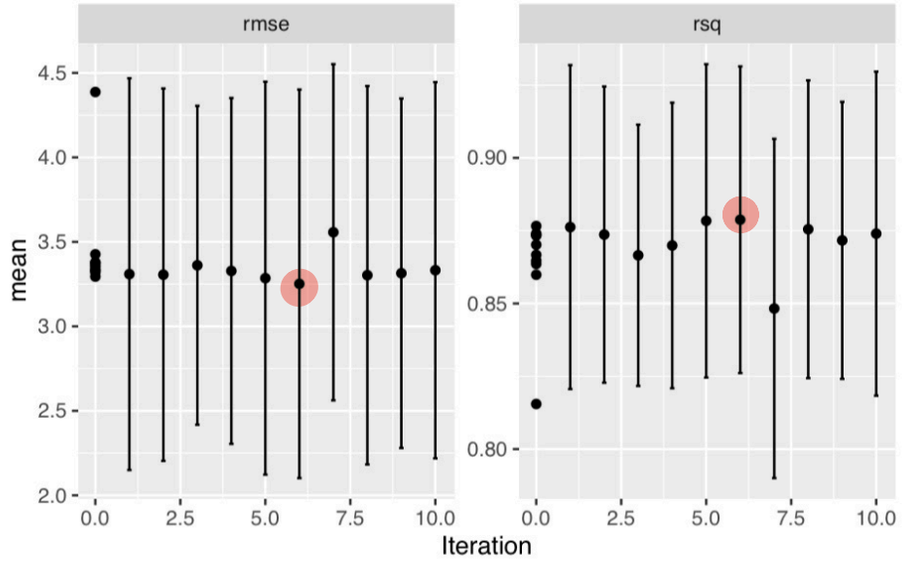

rmse
rsq
mean
5
4
3
2
0.90
0.85
0.80
0.75
0.0
2.5
5.0
7.5
10.0
Iteration

mtry

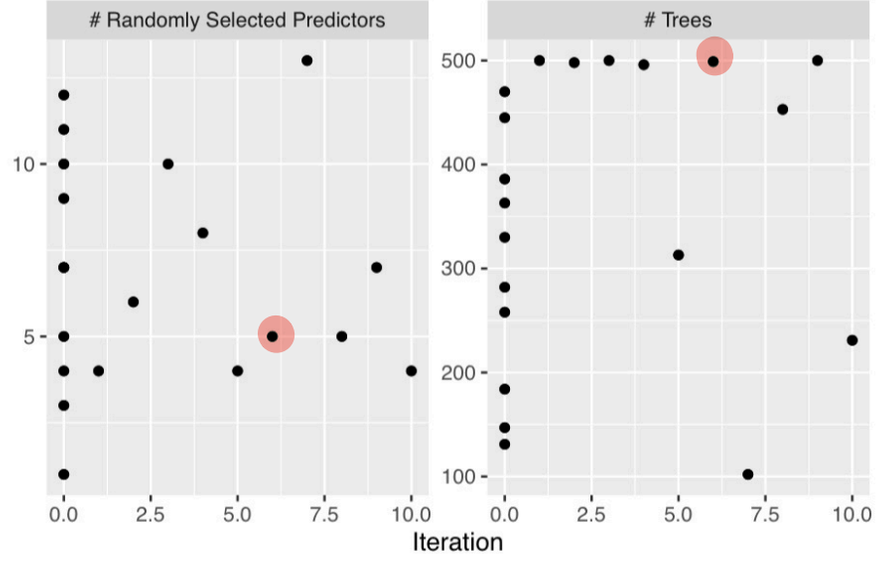
B


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

```
# 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     4   333 rmse    standard    3.26     5   0.439 Preprocessor1_Model~     0
 2     6   423 rmse    standard    3.26     5   0.416 Preprocessor1_Model~     0
 3     4   500 rmse    standard    3.28     5   0.442 Iter4                    4
 4     5   336 rmse    standard    3.29     5   0.446 Iter1                    1
 5     6   347 rmse    standard    3.29     5   0.411 Preprocessor1_Model~     0
 6     6   500 rmse    standard    3.29     5   0.413 Iter6                    6
 7     7   500 rmse    standard    3.30     5   0.411 Iter3                    3
 8     8   399 rmse    standard    3.32     5   0.393 Preprocessor1_Model~     0
 9     9   186 rmse    standard    3.33     5   0.367 Preprocessor1_Model~     0
10     9   477 rmse    standard    3.34     5   0.384 Preprocessor1_Model~     0
```




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



```
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     5   499 rmse    standard    3.25     5   0.447 Iter6                    6
 2     4   313 rmse    standard    3.29     5   0.452 Iter5                    5
 3     7   445 rmse    standard    3.29     5   0.399 Preprocessor1_Model~     0
 4     5   453 rmse    standard    3.30     5   0.436 Iter8                    8
 5     6   498 rmse    standard    3.31     5   0.429 Iter2                    2
 6     4   500 rmse    standard    3.31     5   0.451 Iter1                    1
 7     7   500 rmse    standard    3.31     5   0.402 Iter9                    9
 8     5   258 rmse    standard    3.32     5   0.416 Preprocessor1_Model~     0
 9     8   496 rmse    standard    3.33     5   0.398 Iter4                    4
10     4   231 rmse    standard    3.33     5   0.433 Iter10                  10
```

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

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.

Bayesian Optimization is one way to optimize expensive functions

Assume a Bayesian prior on F
(usually a Gaussian process prior)

while (budget is not exhausted) {

Find x that maximizes acquisition(x,posterior)

Sample x & observe F(x)

Update the posterior distribution on F

}

[Slide from talk by Frazier]

Design of experiments and response surface methodology

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.

See separate slide-deck made by Håkon Gryvill, Yngvild Hamre and Javier Aguilar for the article presentation in MA8701 in the spring of 2021.

NTNU | Norwegian University of Science and Technology

DESIGN OF EXPERIMENTS AND RESPONSE SURFACE METHODOLOGY TO TUNE MACHINE LEARNING HYPERPARAMETERS, WITH A RANDOM FOREST CASE-STUDY

Article presentation in MA8701 by Javier, Håkon and Yngvild

Introduction - Idea in this paper

1. Find most important hyperparameters (factors) in the random forest algorithm using design of experiments (DOE)

2. Apply response surface methodology (RSM) on the parameters chosen in step 1

Background - Design of experiments (DOE)

A response variable may be impacted by controllable and uncontrollable factors.

- **Controllable factor:** The experimenter can freely alter its levels.
- **Uncontrollable factor:** Variables that are not controlled by the experimenter, but can be monitored and even included in the model.

Background - Design of experiments (DOE)

Principles of DOE:

1. **Randomization:** experiments should be run in a random order to prevent external factor from affecting results.
2. **Replication:** allows calculation of internal s.e
3. **Blocking:** can reduce variability

Background - Design of experiments (DOE)

TMA4267

Two level factorial design (2^k):

- Most basic type of experiment.
- k factors at two levels: low and high.
- Regression model:

low high

2·2

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i<j} \beta_{ij} x_i x_j + \varepsilon$$

where $\beta_i, i = 1, ..k$ are main effects and $\beta_{ij}, j = 2, ..., k$ are interaction terms.

As k increases, the number of runs increases exponentially.

Idea: use a fractional DOE

Background - Design of experiments (DOE)

Fractional Factorial DOE (2^{k-p}):

1. Fewer runs are needed 2^{k-p}.
2. Trade-off: loss of accuracy due to fewer df to evaluate each factor and every possible interaction.
3. Powerful screening methods. Usually done at the beginning of experiment to see which factors are important.

Background - Design of experiments (DOE)

3 unique characteristics that make them highly efficient:

1. **Sparsity of effects principle:** only a small number of effects are significant and the final model is composed of low order terms.
2. **Projection property:** a design can be projected into a lower dimension using a subset of factors.
3. **Fold over:** FFDOE can be combined to form designs of higher resolution. Helps in isolating main effects.

Serious disadvantage of FFDOE: unable to detect quadratic effects.

Background - Response Surface Methodology

RSM: Procedure used to model a surface using statistical techniques for the purpose of optimizing a response.

Objective: Find value of x that maximizes response y, with

$$y = f(x) + \varepsilon,$$

where ε is the error and the response surface is $\eta = f(x)$.

Challenge: a priori f is an unknown function.

Background - Response Surface Methodology

Methodology: find a model which fits the relationship between the predictors and the response using a polynomial function.

Popular choices:

- **First-order model:**

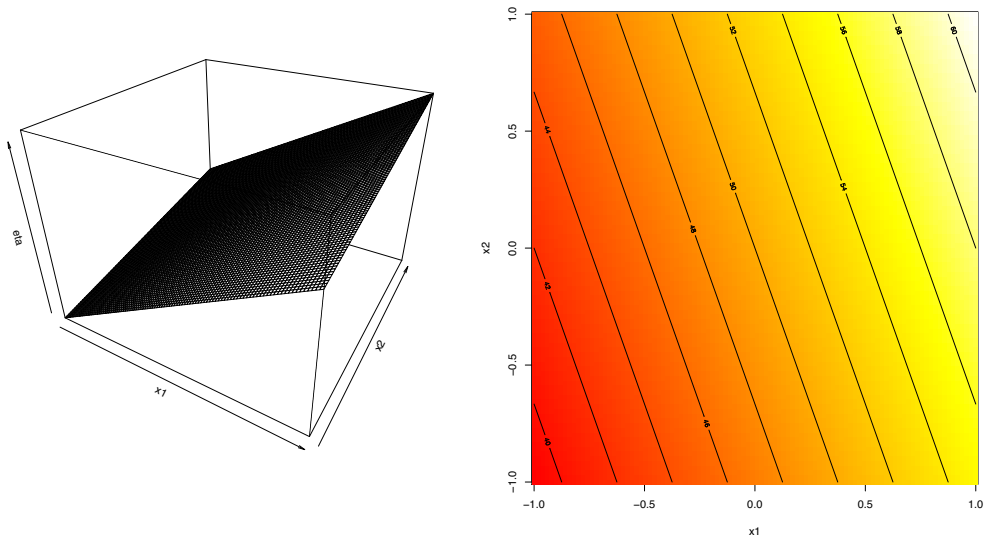
$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \varepsilon$$

- **Second-order model:**

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{jj} x_j^2 + \sum_{i<j} \beta_{ij} x_i x_j + \varepsilon$$

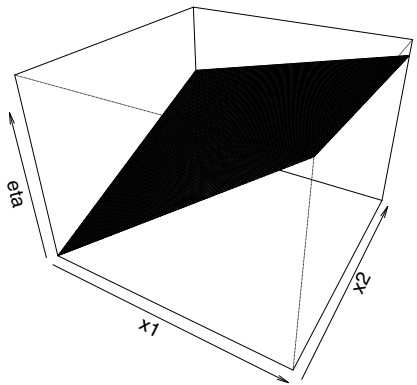
Background - Response Surface Methodology

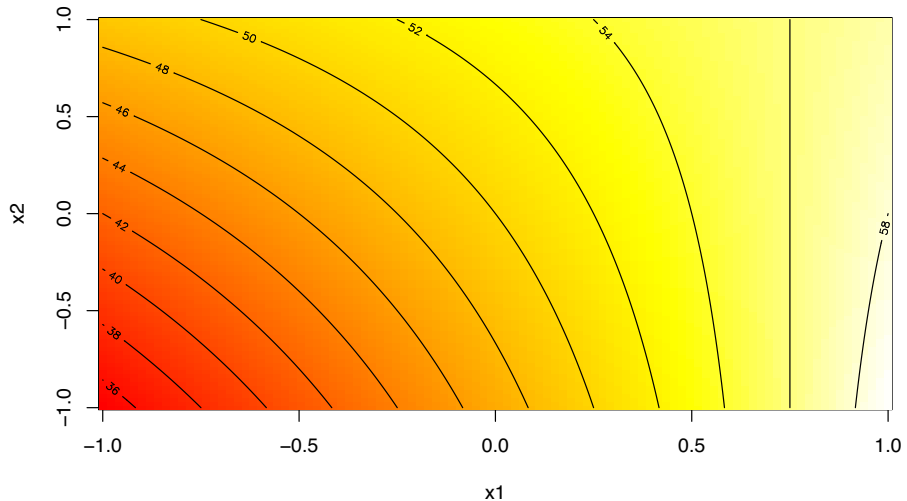
Main effects



Background - Response Surface Methodology

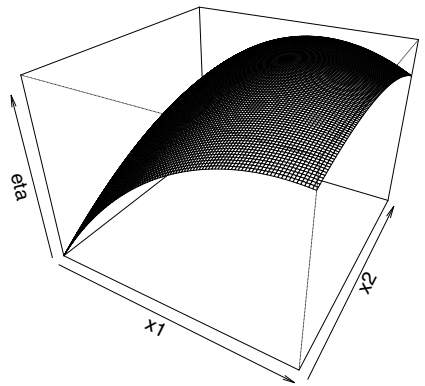
Interaction

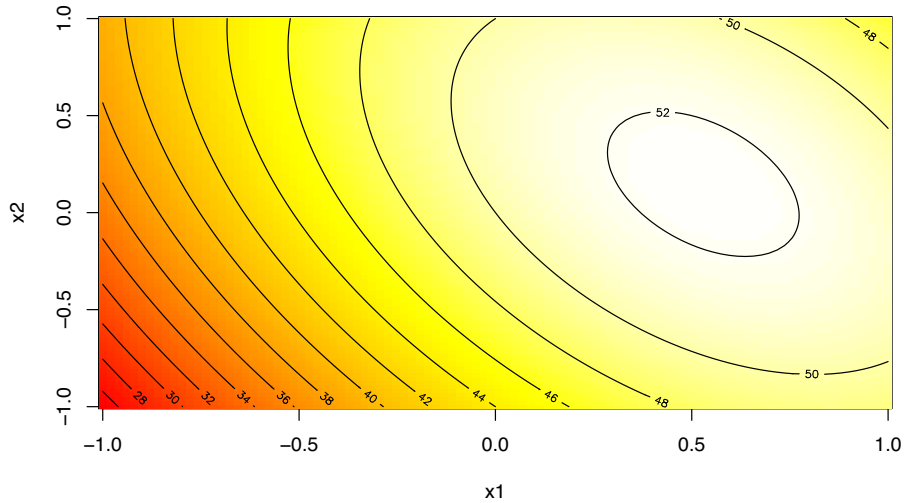




Background - Response Surface Methodology

Quadratic





Background - Response Surface Methodology

- RSM is **sequential procedure** where at each step, we move in a direction of improvement for our objective.
- **Steepest ascent** or ridge analysis is used to move to optimal region or the response surface.
- The procedure is repeated until no more improvements are found in a local neighborhood.

Most popular RSM designs: Central Composite designs (CCD) , Box-Behnken (BBD)

Background - Performance metrics

Balanced accuracy (BACC):

$$BACC = (TPR + TNR)/2, \tag{1}$$

where $TPR = TP/(TP + FN)$ and $TNR = TN/(TN + FP)$.
Good metric for highly unbalanced data

Background - Random forest

Bagging: create B bootstrap samples and fit a decision tree to each sample
Random forest: in each split, we are only allowed to consider m of the p predictors
Use fully-grown trees rather than pruned ones
$\Longrightarrow$ Less correlated

Background - The `RandomForest` package in R

Hyperparameters in `RandomForest`:

1. ntree: number of trees to grow
2. mtry: number of predictors m allowed to be considered at each split
3. replace: should sampling be done with our without replacement?
4. nodesize: minimum size of leaf nodes
5. classwt: prior probability for each of the classes
6. cutoff: threshold for binary classification
7. maxnodes: maximum number of leaf nodes a tree can have

$d = 7$

Experiments - The dataset

Aim: classifying whether a person makes over 50 000 USD per year
32561 observations, 14 covariates
Some of the covariates:

1. age
2. marital status
3. race
4. sex
5. education

Experiments - General procedure for hypertuning using DOE and RSM

Procedure

1. Choose a machine learning algorithm and decide on the response variable to tune (accuracy, TPR, F1-score, etc.)
2. Select the hyperparameters to tune as well as their ranges
3. Perform a screening design and identify the important hyperparameters
4. Reduce the model and, depending on the number of experiments that are feasible to run, perform either a full or fractional 2k factorial design
5. Fit a second-order model using RSM (CCD, BBD), selecting the hyperparameter configuration with the best performance from the previous step as the center of the design
6. Recursively optimize the second-order model until the change in the response is $\leq \epsilon$.

Experiments - Comments to the procedure

- Throughout each of these steps, the response variable should be estimated using n-fold cross-validation.
- The result of the procedure will be compared to the default settings
- The data set is small enough to accommodate a full factorial as the first run, but they choose to pretend that initial screening is needed
- The initial screening is performed using a 2^{7-2} design, so some two-factor interactions are confounded

Experiments - Initial levels for screening

Table: Factors and levels in the initial screening

Factor	Low factor level (-)	High factor level (+)
ntree	100	500
mtry	2	4
replace	FALSE	TRUE
nodesize	1	3256
classwt	1	10
cutoff	0.2	0.8
maxnodes	5	NULL

Experiments - Analysis of first screening

Coefficients	Estimate	Std. Error	*t*-value	$P(>\|t\|)$
(Intercept)	0.3458	0.0043	80.503	2.47E-10 ***
ntree	0.0029	0.0043	0.684	0.5193
mtry	–0.0069	0.0043	–1.614	0.1578
replace	–0.0253	0.0043	–5.879	0.0011 **
nodesize	0.0435	0.0043	10.132	5.37E-05 ***
classwt	–0.1364	0.0043	–31.766	6.47E-08 ***
cutoff	0.0475	0.0043	11.07	3.24E-05 ***
maxnodes	–0.0593	0.0043	–13.816	8.95E-06 ***
ntree:mtry	–0.0371	0.0043	–8.636	0.0001 ***
ntree:replace	0.0003	0.0043	0.085	0.9357

- Confounded effects significant, need follow-up. Use fold over design.

Experiments - Analysis of second screening

Coefficients	Estimate	Std. Error	*t*-value	$P(>\|t\|)$
(Intercept)	5.92E-01	7.82E-03	75.777	2E-16
ntree	–9.07E-04	7.82E-03	–0.116	0.9082
mtry	5.36E-03	7.82E-03	0.686	0.4975
replace	1.61E-03	7.82E-03	0.206	0.8377
nodesize	–6.41E-03	7.82E-03	–0.821	0.4174
classwt	–1.42E-02	7.82E-03	–1.818	0.0777 +
cutoff	–3.06E-03	7.82E-03	–0.391	0.6978
maxnodes	1.39E-02	7.82E-03	1.782	0.0834 +
ntree:mtry	4.29E-04	7.82E-03	0.055	0.9566
ntree:replace	–3.16E-03	7.82E-03	–0.405	0.6882

- Significant two-factor interactions: The hierarchy and heredity dilemma

Main results - Initial screening

- ntree not significant - saving computations by setting it low
- Note: A hyperparameter not being significant in this particular case can matter in other settings
- Having identified the active factors, a full factorial experiment was conducted
- Results analyzed, maxnodes removed, new full factorial with factors nodesize, classwt and cutoff

Main results - RSM for optimization

- Having completed the screening phase, it was time to optimize
- Used Box Behnken design, suited for fitting second-order models (several levels for each factor)
- Fitted model, found the significant terms, fitted reduced model
- Steepest ascent, but not outside the experimental region
- New experiment, new model and new steepest ascent
- Satisfying results - 0.81 in BACC compared to the default 0.64

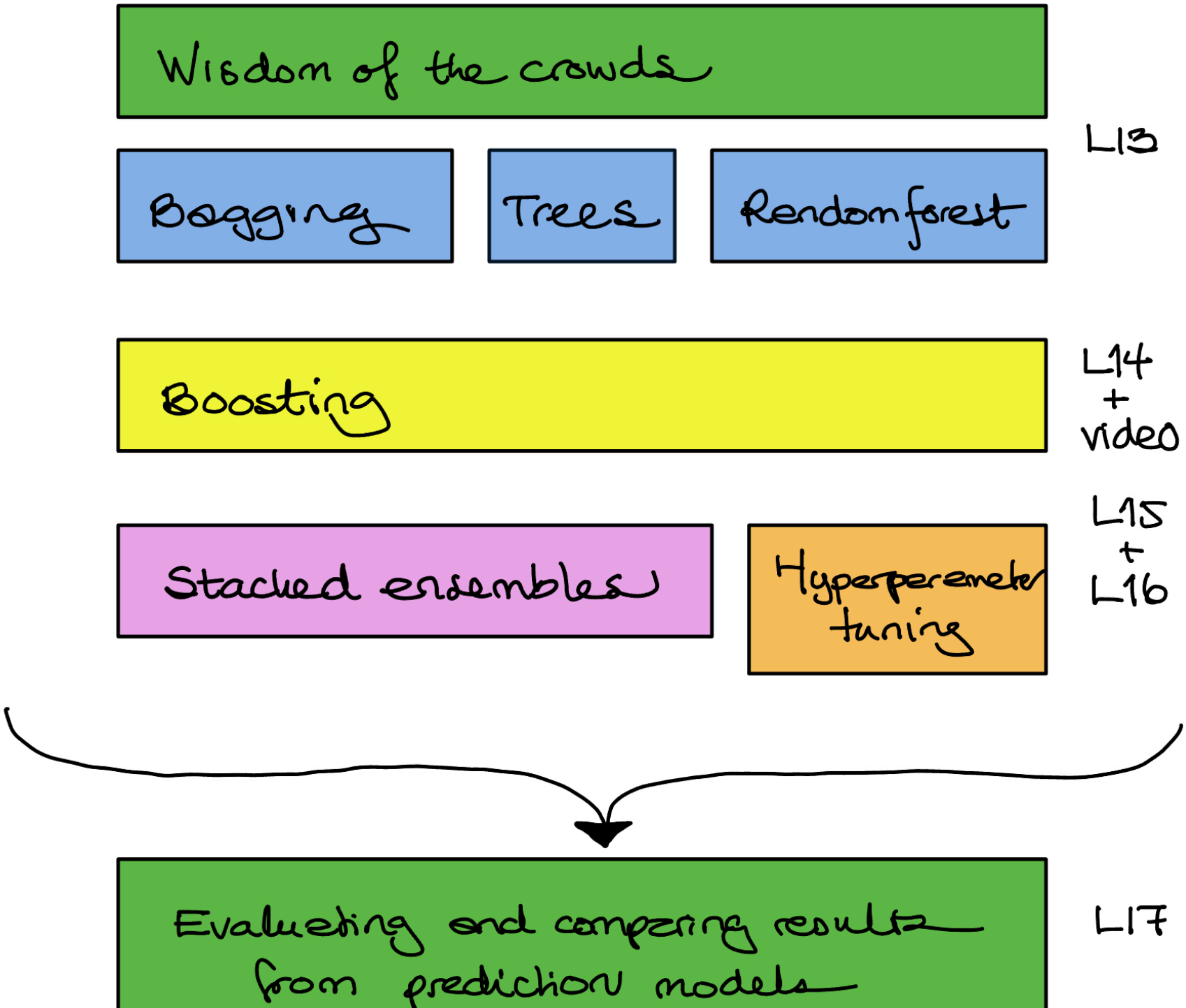
Discussion and conclusion - part 1

- Saving computations by using low levels of hyperparameters that are not significant
- Some parameter can compensate for each other
- Method allows us to understand which hyperparameters matter and how they impact the result - but the spesifics do not necessarily generalize
- Convexity unrealistic - probably found local maximum

Discussion and conclusion - part 2: Our comments

- Advantages of the method: Can save computation and gain information about which hyperparameters matter
- Disadvantage: Not possible to use this if very many hyperparameters must be tuned. Requires a lot of domain knowledge. Should probably be automated to achieve popularity
- Would have been interesting: Comparison with grid search and Bayesian optimization
- More information about computational demands
- Confidence intervals for BACC

Before we start



References

- M. Feurer and F. Hutter (2019). In F. Hutter et al (eds.) Automated Machine Learning. The Springer Series on Challenges in Machine Learning.
- Jo Eidsvik (2017): Introduction to Gaussian processes, note to TMA4265.
- Peter I. Frazier (2018): A tutorial on Bayesian optimization. arxiv https://arxiv.org/abs/1807.02811
- Max Kuhn and Julia Silge Version 0.0.1.9008 (2021-02-15) Tidy modelling with R. https://www.tmwr.org/
- Roger Gosse, University of Toronto: CSC321 Lecture 21: Bayesian Hyperparameter Optimization. http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec21.pdf
- Max Kuhn (2020). caret: Classification and Regression Training. R package version 6.0-86. https://CRAN.R-project.org/package=caret