Supervised Machine Learning HW 3

Group 3: Maja Nordfeldt, Jakob Rauch, Timo Schenk, Konstantin Sommer

25-1-2020

Introduction

How well can we predict an airline's revenue passenger miles (RPM) in a given year using fuel price and airline characteristics? RPM indicates how many miles an airline transports its paying customers and is a widely used output measure in the aviation industry. For airlines' financial and stratgic planning, the prediction problem is therefore of obvious importance.

However, the problem presents a number of practical challenges: First, we only have a very limited set of predictors available. To complicate things, there might be two-way or even three-way interactions between the predictor variables that significantly affect output. Finally, there might be important nonlinearities in the relation between our predictors and output: For instance, when fuel prices rise above a certain threshold the airline might find that servicing some market segments has become entirely unprofitable and therefore reduce its operations. Then the effect of fuel price on RPM is limited for low fuel prices but large for already high fuel prices.

To deal with these problems, we use the method of kernel ridge regression (KRR). In the following, we will give an overview of our data, explain our method and modeling choices in detail, and discuss our results.

Data

We have a balanced panel of six airlines spanning the years 1970 to 1984. The data was originally used by Greene et al. (1997) and is supplied with the R package Ecdat. The panel contains 4 variables: fuel price, total costs of the airline in 1,000$, load factor (the average capacity utilization of the fleet) and output in RPM (as already explained). All of these variables are specific to a given airline. This seems to be true even for the variable fuel price, perhaps indicating that airlines get different fuel prices as a result of quantity discounts. We are only speculating,however, as we do not have a detailed description of this variable.

Table 1 gives an overview of the differences between airlines. As we can see, the airlines vary wildly in size, with airline 1 having more than ten times the output of airline 2. All airlines do seem to pay roughly the same price for fuel, although the bigger airlines seem to benefit from a small quantity discount. Bigger airlines also seem to do better in terms of load, i.e. utilizing their fleet to the fullest. This relation, however, does not hold in every case.

Table 1: Airline averages over all years

airline	Cost	Output	Fuel_Price	Load_Factor
1	2639003	1.41	462320	0.5972
2	2127884	1.017	466287	0.5471
3	723189	0.4124	476349	0.5845
4	638088	0.2126	473368	0.5477
5	293276	0.1138	478079	0.5665
6	313702	0.1043	473695	0.5198

Table 2 gives an overview of the time trends across all airlines. Clearly, output has grown a lot over the 15 years, with only a short interruption around 1980. Costs have grown even stronger, although this might be

due to inflation Again, we do not know for lack of a detailed variable description, but if the costs are really not inflation adjusted this is a limitation of the data. Fuel prices, on the other hand, have fallen a lot over the years, indicating perhaps the recovery from the oil crisis. Finally, the airlines seem to have improved their efficiency over the years and have a 10% higher load factor at the end of the sample period.

Table 2: Yearly averages over all airlines

year	Cost	Output	Fuel_Price	Load_Factor
1	383623	0.3185	113259	0.4789
2	417686	0.3295	116954	0.4868
3	476467	0.3826	117661	0.5236
4	568154	0.43	124606	0.5244
5	656984	0.4413	213491	0.5635
6	725930	0.4475	279172	0.5542
7	808122	0.4894	305404	0.5607
8	924717	0.5289	351094	0.5671
9	1075791	0.622	385088	0.6179
10	1336726	0.6985	557924	0.6234
11	1610212	0.6422	858157	0.5803
12	1772422	0.6333	1007688	0.5856
13	1861304	0.6694	945855	0.5803
14	2022234	0.7489	868171	0.5805
15	2197484	0.793	830720	0.5797

Note that from the original dataset we transform all variables to z-scores, removing their unit of measure. After this transformation, each variables value can be interpreted as standard deviations from its mean. The importance of this will be explained in the next section. Additionally we will transform the panel data set to a cross sectional one, by transforming all categorical variables, airline and year, into dummys. Such that we have one dummy per airline and one per year. Their coefficients could in an OLS context be interpreted as fixed effects.

Methods

We employ two types of *kernel ridge regression* for our predictions. In general, a kernel ridge regression combines the penalty of ridge regression to avoid over-fitting, with a set of non-linear prediction functions in a computationally simplified way to allow for model complexity. This means the method handles issues tied to the trade-off between model complexity and variance well. For example, when opting for a more complex, non-linear model over a simpler option, one is often still limited in what complexity level that is feasible and can produce adequate forecast precision due to the uncertainty in model choice and parameter estimation. Meanwhile, a high number of predictors can provide a better fit, but risk to worsens predictions if observations are sparse, why a reduction is desirable. Kernel ridge regression handles both of theses concerns, and is particularly attractive for estimating non-linear models of "fat data", where the number of predictors greatly exceed the number of observations.

With kernel ridge regression, we map predictors to a high-dimensional space of nonlinear functions, and in this space the forecast is estimated, with the penalty term counteracting overfitting. However, by setting the kernel in a convenient way, calculations need not to be done in the high-dimensional space, but in a simpler manner as will be explained later.

More precisely, for each observation i, we want to predict y_i with $\mathbf{x_i}$, and seek a function of $f(\mathbf{x_i})$ which approximates y_i well for all our observations. We will choose $f(\mathbf{x})$ from a set of linear combinations of the elements of $\phi(\mathbf{x})$, a function mapping the data from the input data dimension space p to one of higher

dimensions. With a large number of regressors after the mapping, the functional form will be very flexible, but computationally effortsome. With a kernel function $K()$, where $K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^T\phi(\mathbf{x_j})$, representing a dot product in the higher dimensional space, we can find the dot product under p-dimensional computations. This means we avoid explicitly transforming each data point with ϕ, but can instead represent them with pairwise values - the kernel values of pairs of points. In other words, instead of mapping points into the higher dimensional space, the data is represented by the kernel matrix $\mathbf{K}$, over which all the computations can be done. The effective dimensions are less than the ones introduced by the kernel, as the basis functions are shrinked. Kernels can also be thought of as functions measuring how closely related the input vectors are - if they are similar the kernel output will be large, and dissimilar it will be small.

Decisions of the method include choosing the kernel and hyper parameter values. The kernel choice will determine the functional form, and setting tuning parameters can require a trade-off between effortful cross-validation or non-optimal results. Here we have chosen two commonly used kernels and used a combination of cross-validation and best practice for setting our hyperparameters. As in the case of a ridge regression it is essential to before analysis standardize our variables to z-scores such that they will be penalized on an equal basis, which would not be the case if the variables would exhibit different scales and variances.

The first kernel we use is the *radial basis function kernel*, for which we have $k_{ij} = exp(\gamma||\mathbf{x_i} - \mathbf{x_j}||^2)$, with $\gamma > 0$. If $\gamma = 2\sigma^{-1}$, it is referred to as the *gaussian kernel*. Generally, this kernel corresponds to mapping of data to an infinite dimensional space. Observations are symmetrically centered by a radial basis function, and the γ parameter sets the spread of the kernel. A larger γ implies more local kernel functions, increasing the effective dimensions and capturing more complexity and shape of the data. We set γ to $\frac{1}{p}$, as commonly used in the literature.

The second kernel we employ is the *inhomogeneous polynomial kernel*, under which we can do a polynomial regression with high-order polynomials in a very short amount of time. Here we have $k_{ij} = (1 + \mathbf{x_i}^T\mathbf{x_j})^d$, for a degree $d > 0$. The degree determines the highest polynomial degree assigned to a predictor, and translates into how many basis functions that are created - $\binom{d+p}{d}$ - each basis function representing the predictors at polynomial degree up to d and their interaction with other observations. In the higher dimensional space, we can use linear methods, while exploring non-linear patterns and relationships as the terms are non-linear transformations of the original data. As already stated in the introduction we assume that there might be interactions between our variables that can help us predict our endogenous variable. For example we assume that there might be synergy effects between the fuel price of a company and the load factor in the sense that cheap fuel might in combination with a high load factor further improve the airline outcome on top of the two coefficients themselves.

By using the so called dual approach, we will not be minimizing over our actual coefficients but over $\mathbf{q}$, which is defined as the predicted values $\mathbf{Xw}$, where $\mathbf{w}$ are the regression weights. We are using the analytical formula provided in the slides and replace the Ridge Kernel by the above described ones such that: $\tilde{\mathbf{q}} = (\mathbf{I}+\lambda\mathbf{K^{-1}})^{-1}\mathbf{y}$, where we oppress the $\mathbf{J}$ matrix since our variables are already standardized and therefore demeaned.

The penalty term λ will be determined in a k-fold cross validation together with either γ or d. For this method, we split our data sets into training and test data and use our training data to estimate $\tilde{\mathbf{q}}$ with different choices for the hyperparameters. We then try to predict the outcome of the test data set and estimate the root mean squared error (RMSE) for every choice of the hyperparameters. For the prediction we use the formula $\mathbf{q_u} = w_0\mathbf{1} + \mathbf{K_u}(\mathbf{K} + \lambda\mathbf{I_n})^{-1}\mathbf{J}\mathbf{y}$[1], where $\mathbf{q_u}$ denotes predicted values and $\mathbf{K_u}$ the kernelized test data. We add a term to $\mathbf{K}$ to ensure invertability. We then vary test and trainings data set and choose in the end the hyperparameters that minimize "on average" the RMSE.

Results

Figure 1 shows the results from our own 9-fold crossvalidation of λ. Due to the random reshuffling of our data, the optimal λ varies. For the Gaussian kernel with fixed value of $\gamma = \frac{1}{p}$, with p the number of predictorsis, λ

[1] as proven in lecture 3, slides p57-58

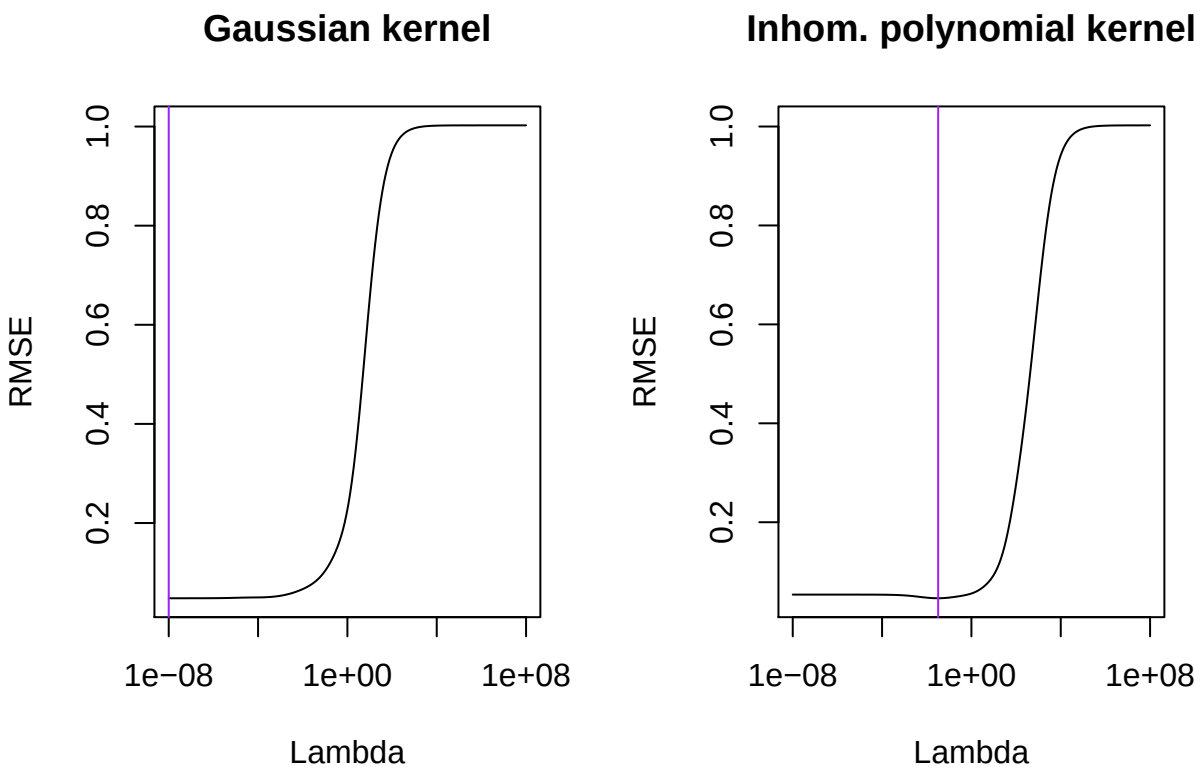

Figure 1: Results from manual 9-fold cross validation

is usually around 0.0003. For the inhomogenous polynomial kernel, the optimal λ is 0.05. For both kernels the resulting RMSE is around 0.05, which we can interpret as an R^2 of 0.95 since all our data (including y) is standardized.

Figure 2 shows the results from the DMSLE package. As we can see, the optimal λ values are roughly the same. However, the values and pattern in RMSE are different in the package. We have not been able to find what causes this, there seems to be some issue with the scale of our RMSE, considering that we still get the same optimal value.

Following the KRR method, we do not get a vector of weights or point estimates indicating which variables have important effects. We can merely make statements about predictive performance, which is very good.

Conclusion & Discussion

In this paper we wanted to investigate how well we could predict an airline's revenue passenger miles (RPM), using fuel price and airline characteristics. Using kernel ridge regression, we found that both the radial basis function kernel and the inhomogeneous polynomial kernel produced relatively accurate predictions.

However, the analysis is sensitive to the decisions we have made - we use two particular kernels because of their inherent benefits, but it is not clear how to optimally select a kernel and by our choice we introduce a functional form preference. We tune one of our hyperparameters and choose the other, meaning there is as a risk of suboptimal results.

Although we achieve predictions, we are in this analysis not distinguishing between what predictors are important, their relative importance and the direction of that importance. Therefore, we cannot make any statements about the distinct explanatory factors. Nevertheless, predictions can still be of use to planners in the aviation industry for their prospective business decisions. It may also be of interest to policy makers, as

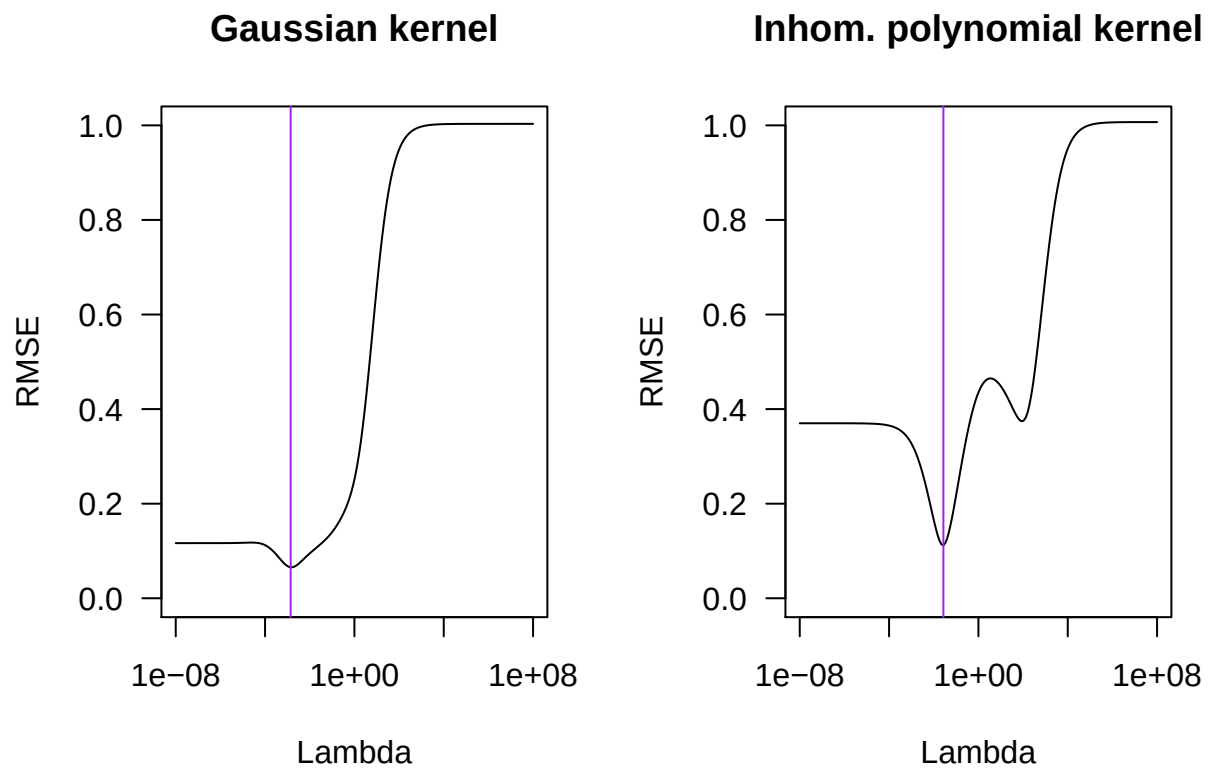

Figure 2: Results from package 9-fold cross validation

an increased taxation on fuel would likely influence airlines profitability and this could be a way to understand how businesses would react.

References

Greene, William. "Frontier production functions, M. Hashem Pesaran and Peter Schmidt (eds.): Handbook of Applied Econometrics, vol. II." (1997): 81-166.

Code

```
# *------------------------------------------------------------------
# | PROGRAM NAME: Supervised Machine Learning - HW 3
# | DATE: 23-1-2020
# | CREATED BY: Jakob Rauch
# *------------------------------------------------------------------

# Clear cache
rm(list = ls())

# Load packages
library(dsmle)
library(plyr)
library(pander)
```

```
est.qtilde.gauss = function(y,K,lambda){
  # input: nx1 vector y (mean zero), nxn kernel matrix (K),
  #         tuning parameter lambda
  # return: Kinvq, nx1 vector of in sample predictions minimizing kernel rifge regression loss
  #         multiplied by K^-1

  n = dim(K)[1]
  Kinvq = solve(K+lambda*diag(n))%*%(y-mean(y))

  return(Kinvq)
}


av.rmsfe = function(y,X,fold,gamma,d,lambda){
  # returns average RMSFE for a given lambda over k bins, for both kernels
  n = dim(X)[1] # nr of observations
  k = max(fold) # nr of bins

  # qall.gauss = rep(NA,n) # store out of sample predictions for plot
  RMSFE.gauss = 0
  RMSFE.poly = 0

  for (i in 1:k){
    # split the data
    train = (fold!=i)  # indicator training data
    ytrain = y[train]
    ytest = y[!train]

    # gaussian kernel:
    Kall = phi.gaussian(X,gamma)
    Ktrain = Kall[train,train]
    Ku = Kall[!train,train]
    qnew  = mean(ytrain) + Ku%*%est.qtilde.gauss(ytrain,Ktrain,lambda)
    # qall[!train] = qnew
    RMSFE.gauss = RMSFE.gauss + 1/k*mean((ytest-qnew)^2)

    # inhom. polynomial kernel:
    Kall = K_IHP(X,d)
    Ktrain = Kall[train,train]
    Ku = Kall[!train,train]
    qnew  = mean(ytrain) + Ku%*%est.qtilde.gauss(ytrain-mean(ytrain),Ktrain,lambda)
    # qall[!train] = qnew
    RMSFE.poly = RMSFE.poly + 1/k*mean((ytest-qnew)^2)

  }

    return(sqrt(c(RMSFE.gauss,RMSFE.poly)))
}



cvalkrr.lambda = function(y,X,fold,gamma,d,lambdagrid){
  # returns RMSFE for a grid of lambdas by k fold cross validation
```

```
  RMSFE = matrix(NA,length(lambdagrid),2) # initialize grid of average RMSFE for each lambda

  for (l in 1:length(lambdagrid)) RMSFE[l,] = av.rmsfe(y,X,fold,gamma,d,lambdagrid[l])

  return(RMSFE)
}

## gaussian kernel
phi.gaussian = function(X,gamma) exp(-gamma*as.matrix(dist(X)^2))

## inhomogeneous polynomial kernel
K_IHP = function(X,d) (1+X%*%t(X))^d


# Load supermarket data from github
githubURL = "https://github.com/jakob-ra/Supervised-Machine-Learning/raw/master/HW3/Airline.RData"
load(url(githubURL))
attach(Airline)

y = output
X = model.matrix(~ -1 + factor(airline) + year + cost + pf + lf)[,-1] # gets the X matrix with 5 airlin

# Rescale
X = scale(X)
y = scale(y)

p = dim(X)[2] # Nr of predictors
k = 9 # Nr of folds

lambdagrid = 10^seq(-8, 8, length.out = 200) # values to try for lambda
gamma = 1/p # fix gamma at 1/p
d=2 # fix d at 2



# assign data randomly to one of the k bins
n = length(y)
ntest = floor(n/k) # observations per bin
rank = rank(rnorm(n))
fold = rep(1:k,each = ntest)
fold = fold[rank] # vector of assigned bin

RMSFE = cvalkrr.lambda(y,X,fold,gamma,d,lambdagrid) # matrix of RMSFE for different lambda

min_index = which.min(RMSFE[,1]) # Find index of lowest RMSE
lambda_min_gaussian = lambdagrid[min_index] # lambda value at lowest RMSE

min_index = which.min(RMSFE[,2]) # Find index of lowest RMSE
lambda_min_poly = lambdagrid[min_index] # lambda value at lowest RMSE

# Descriptive statistics
pander(ddply(Airline, .(airline), summarize,  Cost=mean(cost), Output=mean(output), Fuel_Price=mean(pf)
       caption = "Airline averages over all years")
```

```
pander(ddply(Airline, .(year), summarize,  Cost=mean(cost), Output=mean(output), Fuel_Price=mean(pf), Lo
       caption = "Yearly averages over all airlines")
# Plot lambda vs RMSE
op = par(mfrow = c(1, 2))
plot(lambdagrid,RMSFE[,1],type='l', log='x', main = 'Gaussian kernel',
     xlab="Lambda", ylab="RMSE")
abline(v=lambda_min_gaussian, col="purple")
plot(lambdagrid,RMSFE[,2],type='l', log='x', main = 'Inhom. polynomial kernel',  xlab="Lambda", ylab="RM
abline(v=lambda_min_poly, col="purple")
par(op)
# Compare manual to package for RBF
ker.cv.rbf = cv.krr(as.vector(y), X, k.folds = 9, lambda = lambdagrid,
                center = F, scale = F, kernel.type = "RBF", kernel.RBF.sigma = 1)
ker.cv.poly = cv.krr(as.vector(y), X, k.folds = 9, lambda =  lambdagrid,
                     center = F, scale = F, kernel.type = "nonhompolynom", kernel.degree = 2)


op = par(mfrow = c(1, 2))
plot(ker.cv.rbf$lambda,ker.cv.rbf$rmse,type='l', log='x', main = 'Gaussian kernel',
     ylim = c(0, 1), las = 1,xlab="Lambda", ylab="RMSE")
abline(v=ker.cv.rbf$lambda.min, col="purple")

plot(ker.cv.poly$lambda,ker.cv.poly$rmse, type='l', log='x', main = 'Inhom. polynomial kernel',
     ylim = c(0, 1), las = 1, xlab="Lambda", ylab="RMSE")
abline(v=ker.cv.poly$lambda.min, col="purple")
par(op)
# Compare optimal lambda, manual vs package
cat('The optimal lambda for the gaussian kernel is',
    lambda_min_gaussian,
    'according to our manual function and', ker.cv.rbf$lambda.min, 'according to the cv.krr function. Th
    min(RMSFE[,1]), 'and',  min(ker.cv.rbf$rmse), '\n')



cat('The optimal lambda for inhomogeneous polynomial kernels is', lambda_min_poly,
    'according to our manual function and', ker.cv.poly$lambda.min, 'according to the cv.krr function. T
    min(RMSFE[,2]), 'and',  min(ker.cv.poly$rmse), '\n')
```