Capstone Project
Machine Learning Engineer Nanodegree

Demand prediction for a bike sharing systems

July 2016
Philipp Vogler

1. Definition

1.1 Project Overview

I was looking for a problem that is solvable with machine learning in the field of transport and logistics because this is my area of expertise. I found a very promising dataset by the company called Capital Bike Share. It is a bike sharing system in Washington DC. I use this dataset to utilizing machine learning to forecast the demand for the bike sharing system.
I applied different types of regression to find an algorithm to predict the demand for bikes based on calendric and weather information. The information about weather, calendar and bike market is available in a dataset by the University of Porto at UCI ML Repository.
This project tries to create a forecasting function based on two years of historical data by utilizing the machine learning libraries scikit-learn and tensor-flow.

1.2 Problem Statement

For a bike sharing company, the future demand for its bikes is a key indicator to consider when making decisions. Predictions about the demand are vital when scheduling maintenance of the current bicycle fleet or when to acquiring additional vehicles.
The goal of this project is to forecast the demand for bikes in dependency of weather conditions like outside temperature and calendric information e.g. holidays. This information and the demand structure is provided in a set with two years of daily historical data. The demand is given as the total daily demand and as a split for registered users and casual users. To increase the quality of the prediction registered user demand and casual user demand will be predicted separately in step two.
To make predictions machine learning is used to train regressors. Scikit-Learn recommends a support vector regressor (SVR) for this kind of problem and data amount. Also, a deep neuronal network (DNN)

regressor is trained for comparison. To find the hyper-parameters for these regressors GridSearch and RandomizedSearch are utilized. Due to the small dataset cross-validation is applied.

1.3 Metrics

To measure the performance of the regressions, two standard regression metrics are used: Root Mean Squared Error (RMSE) and the coefficient of determination (R^2). Both metrics are calculated for both regressor types. For comparison of the regressors, RMSE is used and R^2 for parameter tuning.

The *RMSE* is the root of the (more common) mean squared error (MSE). The MSE measures the difference between each true value and it's prediction. The Differences are squared to eliminate the signs and to amplify lager differences/errors. Because this error would increase with the number of values, the mean error is calculated over all values. Sometimes the MSE is hard to interpret, because the measurement is also squared and does not relate directly to the measurement of the target values. To counter this problem the RMSE takes the root of the MSE to correct for the squared measurement. "The RMSE is directly interpretable in terms of measurement units, and so is a better measure of goodness of fit than a correlation coefficient." The RMSE is calculated as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where n is the count of observations, y(i) are the true values and y^(i) are the predictions.

R^2 is the coefficient of determination. "It is interpreted as the proportion of the variance in the dependent variable that is predictable from the independent variable." It is the square of the correlation between the true and the predicted values. It ranges from 0 to 1. Higher values show that a lot of variance of the dependent variable is caused by variance in the independent variable. E. g. if the correlation of rainfall and casualty rented bikes is high, the R^2 value is close to 1.

2 Analysis

2.1 Data Exploration

- instant: record index

Feature column(s):
- dteday : date
- season : season (1:springer, 2:summer, 3:fall, 4:winter)
- yr : year (0: 2011, 1:2012)
- mnth : month (1 to 12)
- hr : hour (0 to 23)
- holiday : weather day is holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)
- weekday : day of the week
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
+ weathersit :
    - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
    - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are divided to 41 (max)
- atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)

Target column:
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

Data values:

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit
1	2011-01-01	1	0	1	0	6	0	2
2	2011-01-01	1	0	1	0	0	0	2
3	2011-01-03	1	0	1	0	1	1	1
4	2011-01-04	1	0	1	0	2	1	1
5	2011-01-05	1	0	1	0	3	1	1

instant	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	2	0.344	0.363	0.805	0.160	331	654	985
2	2	0.363	0.353	0.696	0.248	131	670	801
3	1	0.196	0.189	0.437	0.248	120	1229	1349
4	1	0.200	0.212	0.590	0.160	108	1454	1562
5	1	0.226	0.229	0.436	0.186	82	1518	1600

Data stats:

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit
count	731	731	731	731	731	731	731	731
mean	366	2.49	0.50	6.51	0.02	2.99	0.68	1.39
std	211	1.11	0.50	3.45	0.16	2.00	0.46	0.54
min	1.00	1.00	0.00	1.00	0.00	0.00	0.00	1.00
25%	183.5	2.00	0.00	4.00	0.00	1.00	0.00	1.00
50%	366	3.00	1.00	7.00	0.00	3.00	1.00	1.00
75%	548	3.00	1.00	10.0	0.00	5.00	1.00	2.00
max	731	4.00	1.00	12.0	1.00	6.00	1.00	3.00

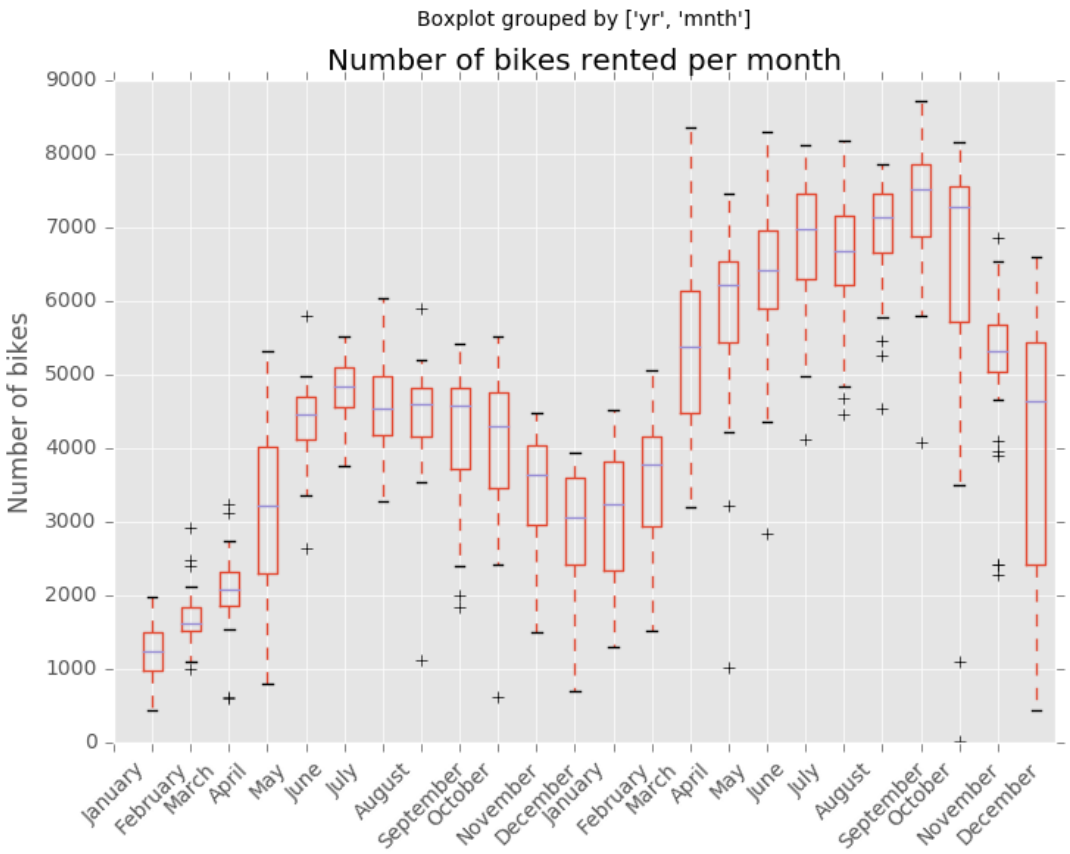
	temp	atemp	hum	windspeed	casual	registered	cnt
count	731	731	731	731	731	731	731
mean	0.49	0.47	0.62	0.19	848	3656	4504
std	0.18	0.16	0.14	0.07	686	1560	1937
min	0.05	0.07	0.00	0.02	2.00	20.0	22.0
25%	0.33	0.33	0.52	0.13	315	2497	3152
50%	0.49	0.48	0.62	0.18	713	3662	4548
75%	0.65	0.60	0.73	0.23	1096	4776	5956
max	0.86	0.84	0.97	0.50	3410	6946	8714

Characteristics

The characteristics of the dataset are very favorable because it was already processed. It is very concise, and missing values are not a problem. Also, most of the data is already normalized or binary. Other categorical data like 'weekday' or 'working day'/'holiday' were processed and transformed into dummy variables.

2.2 Exploratory Visualization

The visualization shows a classic seasonal pattern with an up-trend year over year.
Unsurprisingly bike renting in Washington DC is much more popular in the summer month. Spring and autumn months show higher volatility than the rest of the year, which is likely due to changing weather conditions.
There are some outliers throughout the dataset, mostly on the lower end. These are left in the dataset because they are not due to measurement errors, but to extreme weather conditions. Because extreme weather conditions are part of the problem, the data is not excluded.



2.3 Data Preprocessing (Methodology)

As described in ‘Characteristics’ most of the preprocessing is provided with the data set. Dates get dropped because the regressor can not read this datatype and the order information is already stored in the index. The instant variable replicates this information also. These features are

dropped because the order should not differentiate the data points. The January 1st of 2011 is not better or worse than January 1st, 2012 by the order of the data set. It should differentiate on the year feature, but that information is stored in the ‘yr’ feature already. Keeping date (and instance) in would overemphasize these features.

2.4 Algorithms and Techniques

Creating a model to predict demand from historical data is a supervised learning process. Forecasting the rental bike demand is obviously a quantitative prediction. Therefore a regression algorithm is needed.

A simple *Stochastic Gradient Descent* (SGD) regressor would be the first choice. They are easy to implement and work efficiently. On the other hand, they are “sensitive to feature scaling” and need a lot of iterations and therefore data to provide good results.

Due to the limited number of data points, a more "sophisticated" regressor is necessary. A *Support Vector Regressor* (SVR) is a good choice. They are efficient as well and can operate successfully even if only a limited amount of data (< 10k) is available. Also, they offer different kernels, like a linear or a radial basis function (RBF) kernel and therefore can adapt to the problem at hand. Disadvantages of an SVR are poor performance, “if the number of features is much greater than the number of samples”, which is not the case for this dataset. And they “do not directly provide probability estimates”, which are not necessary for this analysis.

The SVR seems to be the better fitted regressor for the task, especially because of the lower requirements in the count of samples compared with the SGD.

To see, if a *deep neuronal network* (DNN) might be able to beat a linear model for this particular prediction problem the tensor-flow DNN-Regressor algorithm is employed too. DNN Regressors can solve very complex and non-linear problems if sufficiently tuned. Like all artificial neuronal networks, they come with some disadvantages. Neuronal networks (NN) are opaque algorithms, whom parameters and results are hard to interpret. Therefore it was difficult to tune the hyperparameters right. Also, they usually need long training time compared to linear models.

Despite the difficulties with NN tuning, I give the DNN regressor a shot. Maybe the complexity of the dataset is higher than it seems at first glance. Maybe the DNN-Regressor can do at least as good as the SVR. The project is a contest between a support vector regressor and a deep neuronal network regressor.

2.5 Benchmark

Two types of regressors are trained: an SVR and a DNN-Regressor. Both are first used off-the-shelf with default parameters to create a benchmark. Both "benchmarks" for the coefficient of determination are close to zero. The RMSE is pretty significant and similar for both algorithms. Therefore parameter tuning is mandatory for SVR and the DNN Regressor.

```
Score SVR: -0.029012
RMSE SVR: 1895.485129

Score DNN: 0.031714
RMSE DNN: 1838.704961
```

3 Methodology

3.1 Implementation

The regressors are trained using GridSearch and cross-validation to identify the interval of the best parameters. Then a RandomizedSearch is used to fine tune hyper-parameter values of the best results of the regression models.

Therefore the dataset is split randomly into a training set of 75% and a test set of 25% of observations. Furthermore, three-fold cross validation is employed instead of an extra validation set. This choice was triggered by the limited number of observations in the dataset (n=731). Threefold cross-validation is also easy on the computing capacities. Validation is needed due to the iterative nature of the algorithms. Intermediate results for the fit of the hyperparameters are required to tune the properly with gradient decent.

In the first step, GridSearch is used to cover a broad range of values for the C parameter. A factor tree scale is utilized to cover C values (0.1*3

= 0.3*3 = 0.9≈1.0*3 = 3.0 etc.). The other parameter to choose is the kernel. Due to the linear nature of time series data, a linear kernel is expected to work best. But to be safe, a radiant base function (RBF) kernel is tested as well.

After predicting the better kernel (linear) and the best interval for C by grind search, an additional RandomizedSearch is performed. It tries to find the optimum C value in the determined high prospect area of the solution space.

The process worked quite well after the steps were put in the right order. (Using RandomSearch first and GridSearch second leads to inconsistent results for the parameters and lower performance of the model.) GridSearch found parameters that result in significantly better performance. And RandomSearch could increase performance even further, but not much, for both regressors:

SVR tuned with GridSearch and RandomizesSearch

```
Score SVR: -0.029012
Score SVR tuned GS: 0.786930
Score SVR tuned RS: 0.791948

RMSE SVR: 1895.485129
RMSE SVR tuned GS: 862.524552
RMSE SVR tuned RS: 852.306407
```

The tuning works for the SVR.

DNN-Regressor tuned with GridSearch and RandomizesSearch

```
Score DNN: 0.031714
Score DNN tuned GS: 0.131557
Score DNN tuned RS: 0.165697

RMSE DNN: 1838.704961
RMSE DNN tuned GS: 1741.328626
RMSE DNN tuned RS: 1706.758115
```

Same picture with the DNN Regressor. The tuning helps, but the results

are still underwhelming. Also, the best DNN result is no match for the tuned SVR.

Results RMSE

```
SVR tuned: 852.306407
DNN tuned: 1706.758115
```

SVR works much better than the DNN Regressor in comparison.

3.2 Refinement

C influences how smooth the regression line is. The lower the value the smoother the line. In figure 1 it is evident that there are significant seasonal variations in the data, which results in a “bumpy” regression line and therefore very high C values. The linear kernel is the better performing choice. Which is expected due to the linear characteristic of a time series.
The count of rented bikes (cnt) is just the sum of the features casual and registered. Two separate models are trained to predict these features. And add up afterward. Predicting casual and registered users separately should lead to better performance because these customer groups likely have different bike renting patterns. Therefore two separate models should be able to reproduce these behaviors better, than one overarching model. This split should improve the projection.

SVR with GridSearch - for casual users

```
Best parameter from GridSearch: {'kernel': 'linear',
'C': 1000}
```

SVR with RandomizesSearch - for casual users

```
Best CV score from RandomizedSearch: 0.633508
```

SVR with GridSearch - for registered users

```
Best parameter from GridSearch: {'kernel': 'linear',
'C': 3000}
```

SVR with RandomizesSearch - for registered users

```
Best CV score from random  search: 0.794217
```

4 Results

4.1 Model Evaluation and Validation

As mentioned before, there is a 25% hold out for a test set. This data (unknown to the model so far) is used now to measure the performance of the regressors. For the models to generalize well the errors of the test set should not deviate much from the cross validation errors on the last iteration on the training set.

```
Test     set - score cas: 0.634344
Training set - score cas: 0.633508

Test     set - score reg: 0.799791
Training set - score reg: 0.794217
```

The coefficient of determination shows that the model is better in capturing the seasonal and trend effects with the registered users than with the casual users. The improvements over the SVR model that predicts the data set as a whole are marginal:

```
Score SVR - entire set as one: 0.791948
Score SVR -  sum of cas & reg: 0.792683
```

As discussed in the refinement section, we see the linear kernel and pretty high C values for the tuned SVR models, as expected.

SVR for casual users

```
corresponding parameters: {'kernel': 'linear', 'C':
2242.3044446707327}
```

SVR for registered users

```
corresponding parameters: {'kernel': 'linear', 'C':
2732.2436554658093}
```

4.2 Justification

As expected the tuning of the parameters of the regressors improves the

performance. Both regressors make much better predictions after tuning. Parameter tuning with RandomizedSearch can improve the performance even further after the right interval was identified by GridSearch. Despite the tuning, the SVR beats the DNN Regressor by far. The predictions by the SVR are more than five times as good as the DNN Regressors.

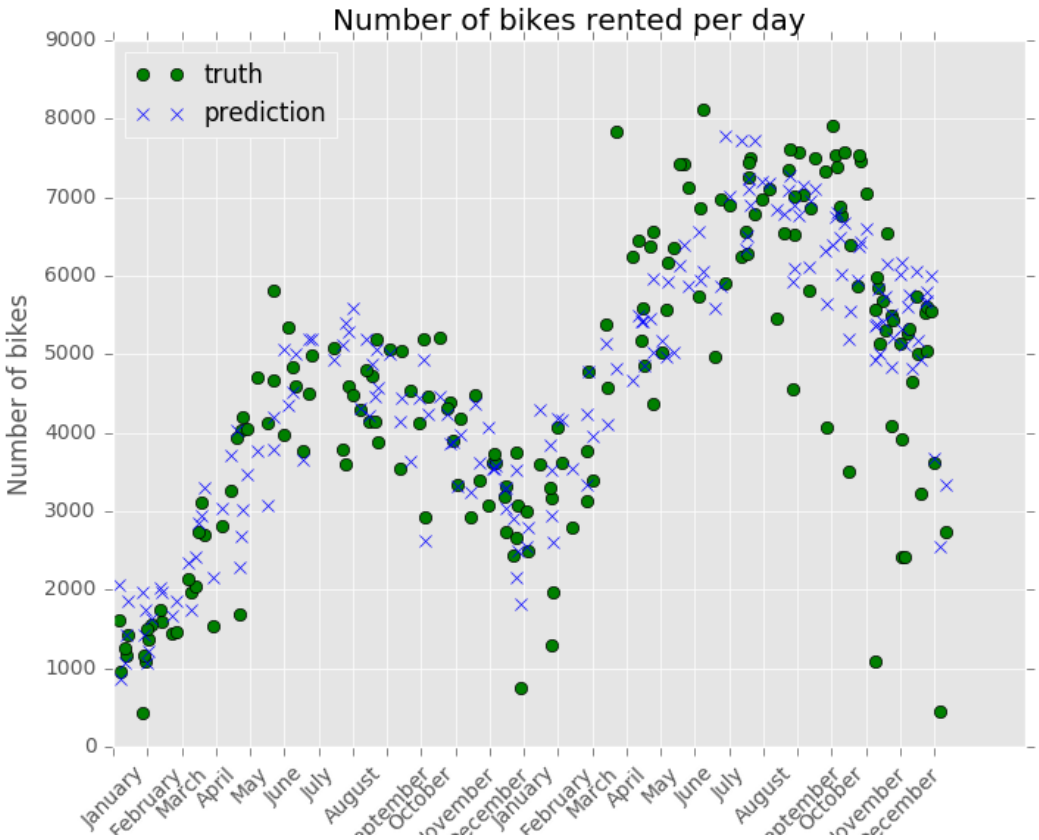
```
Benchmark untuned DNN Score: 0.031714
Tuned DNN Score - entire set as one: 0.165697

Benchmark untuned SVR Score: -0.029012
Tuned SVR Score - entire set as one: 0.791948
2 separat tuned SVR Score -  sum of cas & reg:
0.792683
```

Splitting the dataset and predicting casual and registered customers separately increase the R^2 score also slightly.
A coefficient of determination of almost 80% is a decent result for the SVR regressor. The DNN Regressor predictions are disappointing. Maybe the size of the dataset is insufficient to train the DNN Regressor properly.

5 Conclusion

5.1 Free-Form Visualization



The predictions on the test set are reasonably good, without overfitting. The model captures the upward trend as well as the seasonal curve. About half of the predictions are higher than the truth, and half are lower. As expected from an SVR model, high and low outliers are not captured. The model creates balanced predictions overall.

5.2 Reflection

To predict the demand of the bike sharing company a regression model is developed. A support vector regressor and a deep neuronal network regressor are tested for their performance. For hyperparameter tuning, GridSearch and RandomizedSearch are used with cross-validation on a 75% randomized training set of the provided data. The support vector regressor delivers much better performance. The SVR model is further refined by training one model on just the casual demand and a second separate model on the demand of registered customers. This split increases the performance of the predictive model even further to almost 80% determination.

I had high hopes for the DNN Regressor. It was kind of disappointing that it does not even come close to the SVR results. Maybe my tuning

was not right, or it needs a larger dataset or computational power. Utilizing grid and randomize search in a way that makes sense was a little tricky. At first, the results of the parameter tuning seemed random. This behavior might be caused by local minima in the solution space. Parameter tuning results became more stable when I switched the order of the search methods. It makes more sense to start with a broad GridSearch and then use RandomizedSearch on the given interval, instead of visa verse. It is also computational more efficient.

5.3 Improvement

The coefficient of determination of the regressors could be increased by additional iterations in training and the number of folds in the cross-validation, at the expense of computing time. More iterations might be particularly useful with the performance of DNN Regressor.
The performance of the DNN Regressor might also increase the amount of data available. Of course, there are also other regressors available that might perform better on this particular dataset. For example, a wide and deep learning algorithm might be a better-performing alternative.

Reference

> http://www.capitalbikeshare.com
> http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset
> http://freemeteo.de/wetter/
> http://dchr.dc.gov/page/holiday-schedules
> http://scikit-learn.org/stable/
> https://www.tensorflow.org
> https://www.tensorflow.org/versions/r0.9/tutorials/linear/overview.html#large-scale-linear-models-with-tensorflow
> https://www.tensorflow.org/versions/r0.9/api_docs/python/contrib.learn.html#DNNRegressor
> http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
> http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR
> http://scikit-learn.org/stable/modules/sgd.html#regression
> http://scikit-learn.org/stable/modules/svm.html
> http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html
> http://scikit-learn.org/stable/modules/sgd.html#regression
> http://www.vernier.com/til/1014/
> https://github.com/tensorflow/skflow/blob/master/g3doc/api_docs/python/estimators.md
> http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html
> http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.RandomizedSearchCV.html

> http://scikit-learn.org/stable/modules/generatedsklearn.metrics.mean_squared_error.html#sklearn.metrics.mean_squared_error
> http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2_score
> https://www.kaggle.com/wiki/RootMeanSquaredError
> http://stattrek.com/statistics/dictionary.aspx?definition=coefficient_of_determination
> https://www.reddit.com/r/MachineLearning/comments/3dnolr/disadvantages_of_neural_networksdeep_learning_why/