ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

information
systems

Business Intelligence

– SEMINAR WINTER SEMESTER 2013/2014 –

Support Vector Machines: Introduction and application in forecasting rent prices

– SEMINAR PAPER –

Submitted by:
Mr. Nobody

Student-ID:
xxxxxxx

Advisor:
Prof. Dr. Dirk Neumann

Contents

1 Introduction

A Support Vector Machine (SVM) is a supervised learning algorithm originally developed for two-class classification. However it's properties allowed to apply this method to a multi-class classification, regression and novelty detection, which made SVM quite popular nowadays - it is used widely in medical diagnoses, bioinformatics, text mining, face recognition and image processing.
The method uses simple idea of linear separation of dataset into two subsets. In case if it is not linear separable, it uses kernel tricks to map the data into more dimensional space, where it becomes more separable. Then two parallel hyperplanes constructed in a way to maximize the distance between datasets - the larger distance between them, the less the generalization error of the method will be.
The structure of the paper is as follows: in sections 2 will be given theoretical base for SVM, using this methodology for regression tasks and modifications of original approach. In section 3 we will examine SVM, using rent price data and in section 4 will give conclusions.

2 Support Vector Machines

This section introduces mathematical model of SVM, explains it's concepts and investigates extensions of original approach.

2.1 Basic idea

Key idea of classification in SVM is constructing the optimal hyperplane (or the maximum margin separator) - decision boundary with the largest possible distance to closest points from each class. SVM addresses next issue: instead of minimizing empirical loss on the training set, SVM attempts to minimize expected generalization loss. In other words, it is unknown where future points may fall, but under probabilistic assumption, that they are drawn from the same distribution as the training set, it minimizes generalization loss by choosing the separator that is farthest from already seen points. Points on the boundaries of each class are called support vectors. Now Let's now formalize the model. We have a set of labeled training patterns

$$(x_1,y_1),\dots,(x_n,y_n),\ y_i \in \{-1,1\} \tag{1}$$

which must be linearly separable, if there exists a vector w and a scalar b, such that inequalities

$$w \cdot x_i + b \geq 1, \qquad \text{if } y_i = 1, \tag{2}$$

$$w \cdot x_i + b \leq -1, \quad \text{if } y_i = -1 \tag{3}$$

are valid for all training set.
The optimal hyperplane

$$w_0^T \cdot x + b_0 = 0 \tag{4}$$

is unique one, which separates data with the maximum margin (although there could be infinite amount of non-optimal separating hyperplanes). Each hyperplane is characterized by its exact

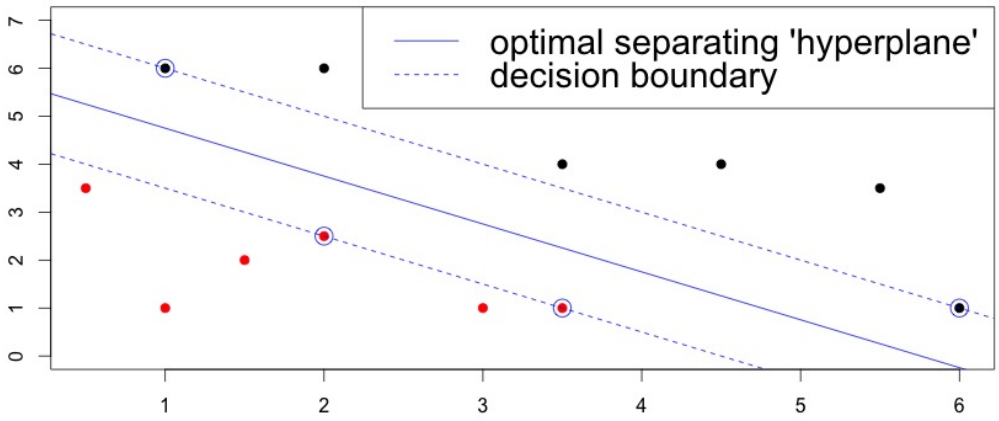

Figure 1: Support Vector Machine. Points of two classes (red and black) are separated by the optimal hyperplane, which position is determined by support vectors - data points with circles around them.

position in space (determined by b) and by direction (determined by $\frac{w}{\|w\|}$), where the distance between the projections of the training vectors of two training classes is maximal. This distance $\rho(w,b;x)$ between point x and optimal hyperplane is

$$\rho(w,b,x) = \frac{w^T x_i + b}{\|w\|}. \tag{5}$$

To construct such optimal hyperplane we need to maximize the distance between projections of training vector of two different classes. This distance $\rho(w,b)$ is given by

$$\rho(w_0,b_0) = \frac{2}{\|w_0\|} = \frac{2}{\sqrt{w_0^T w_0}}. \tag{6}$$

Obliviously, to maximize the distance, we need to minimize $w_0^T w_0$, or more formal by

$$\Phi(w) = \frac{1}{2}\|w\|^2 \tag{7}$$

subject to

$$y_i(w^T x_i + b) \geq 1. \tag{8}$$

Therefore constructing the optimal hyperplane is quadratic programming problem. We solve this optimization problem by introducing Lagrange multipliers, with one multiplier for each constraint, i.e.

$$L(w,b,\lambda) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \lambda_i \Big[y_i(w^T x_i + b) - 1 \Big]. \tag{9}$$

We yield that vector w^*, that determines the optimal hyperplane, can be written as linear combination of vectors x_i, which are not associated with λ_i, such that

$$w^* = \sum_{i=1}^{N} \lambda_i y_i x_i. \tag{10}$$

These are also called *support vectors* and they correspond to the points that lie on the maximum margin hyperplanes. We also should admit that, as it easily seen, optimal hyperplane is independent of number and position of vectors, which are not 'support', what makes this method robust to outliers. Eliminating w and b from equation (9) gives us dual representation of maximum margin problem in which we maximize

$$\bar{L}(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j (x_i x_j) \tag{11}$$

subject to

$$\sum_{i=1}^{N} \lambda_i = 0 \tag{12}$$

$$\lambda_i \geq 0. \tag{13}$$

In dual formulation some algorithms allow us to find solution faster than in primal.

2.2 Soft Margin

Now, let's consider the case, when training dataset cannot be separated without error, for example, if classes are overlapping. This can be solved by introducing the penalty function, a so-called slack variable $\xi_i > 0$, which is the measure of misclassification error.
Our goal is to minimize functional of this error

$$F_\sigma(\xi) = \sum_{i=1}^{N} \xi_i \tag{14}$$

subject to

$$y_i(w^T x_i + b) \geq 1 - \xi_i \tag{15}$$

$$\xi_i \geq 0 \tag{16}$$

And now the goal is to make the margin as large as possible, with, at the same time, the number of points with $\xi_i \geq 0$ as small as possible. Formalizing the task, we introduce positive parameter $C \geq 0$, which controls the trade-off between the slack variable penalty and the margin, or more precise, we need to minimize the cost function

$$\min \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \xi_i \tag{17}$$

subject to constraints equation (15) and equation (16). Solution will be the same as for linear separable case, with only difference in constraint on Lagrangian multiplier $0 \leq \lambda_i \leq C$. In the

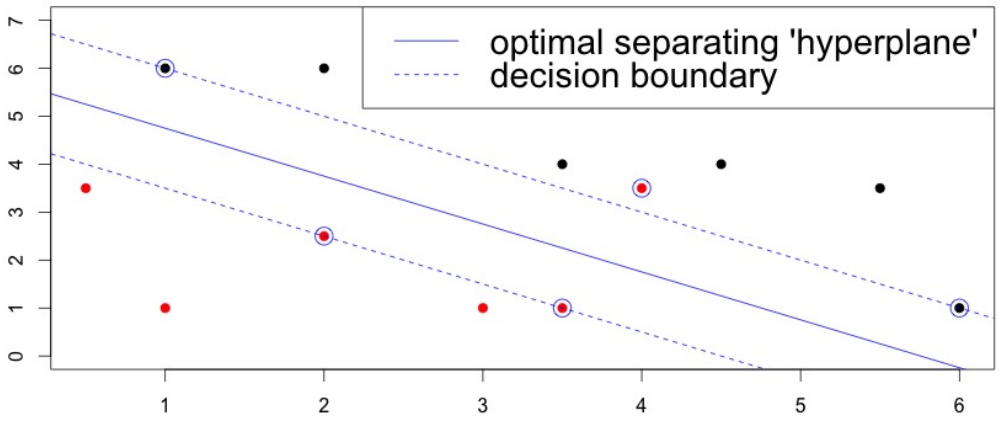

Figure 2: Soft margin, by introducing penalty function, allows classes overlapping. Point with coordinates (4,4), although belonging to 'red' class, lies in the area of 'black' class.

limit $C \to \infty$ we will receive functional for the linearly separable data. This is th case - the margin, associated with the classifier corresponding to the larger value of C, is smaller. This is because the term $C \sum_{i=1}^{N} \xi_i$ has now more influence in the cost, and the optimization process tries to satisfy this demand by reducing the margin and, consequently, by the number of points with $\xi_i \geq 0$. This is the reason why this is also known as soft margin.
Another possible realization is ν-parameterization. Since the margin is such an important entity in the design of SVM (in general, essence of SVM is to maximize it) it is straightforward to include it in more direct way in the cost function, instead of leaving its control to the parameter (i.e. C), whose relation with the margin, although strong, is not transparent. In more general case the margin is defined by pair of hyperplanes

$$w_0^T x + b_0 = \pm\rho \tag{18}$$

where ρ is left as free variable to be optimized. Under this setting optimization problem will look like

$$\Phi(w,\xi,\rho,b) = \frac{1}{2}\|w\|^2 - \nu\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i \tag{19}$$

subject to

$$y_i(w \cdot x_i + b) \geq \rho - \xi_i \tag{20}$$

$$\rho \geq 0 \tag{21}$$

$$\xi_i \geq 0 \tag{22}$$

We see that that the larger ρ the wider the margin and higher the number of points within the margin, for a specific direction w. The parameter ν controls the influence of the second term in the cost function and its value lies in the range [0,1].
ν-SVM has certain advantages comparing to C-SVM. First, it leads to a geometric interpretation

of SVM task for non-separable classes. Second, the constant ν serves as bound for error rate and the number of resulting support vectors.

2.3 Nonlinear case

But in real-world there are a lot of datasets, which are not linearly separable and because of character of the tasks soft margin could not be te solution. To solve this issue let us now assume that there exists a mapping

$$x \in \mathbb{R}^l \to y \in \mathbb{R}^k \tag{23}$$

from the input feature space into a k-dimensional. This transformation gives us an opportunity to separate classes by a hyperplane in a new space. From mathematical view, interesting and important moment is that inner product of the vectors in the new (higher dimensional) space is expressed as a function of the inner product of the corresponding vectors in the original feature space. Such functions are called kernels. Typical examples of used kernels are:

- Polynomial: $K(x,y) = (x^T y + c)^d$.
- Radial basis function: $K(x,y) = \exp(\gamma \|x-y\|^2)$.
- Hyperbolic Tangents: $K(x,y) = \tanh(kx^T y + c)$.

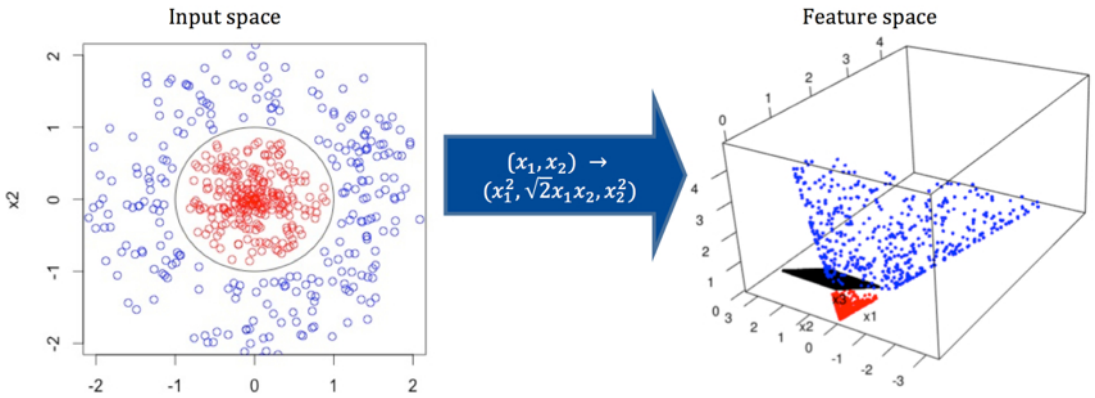

Figure 3: Kernel trick. Mapping from two-dimensional space, with decision boundary $x_1^2 + x_2^2 = 1$, into the feature three-dimansional space $(x_1^2, \sqrt{2}x_1x2, x_2^2)$. Circular decision boundary in original space becomes a linear decision boundary in new space.

2.4 Regression

SVM can also be used for regression problems by introducing of an alternative loss function. The loss function can be modified to include the distance measure. Possible forms of loss function are, for example:

- Quadratic - corresponds to conventional least square error criterion.
- Laplace - less sensitive to outliers.
- Huber - robust loss function that has optimal properties, when underlying distribution data is unknown.
- ε-sensitive - errors lower ε-level are equal to zero.

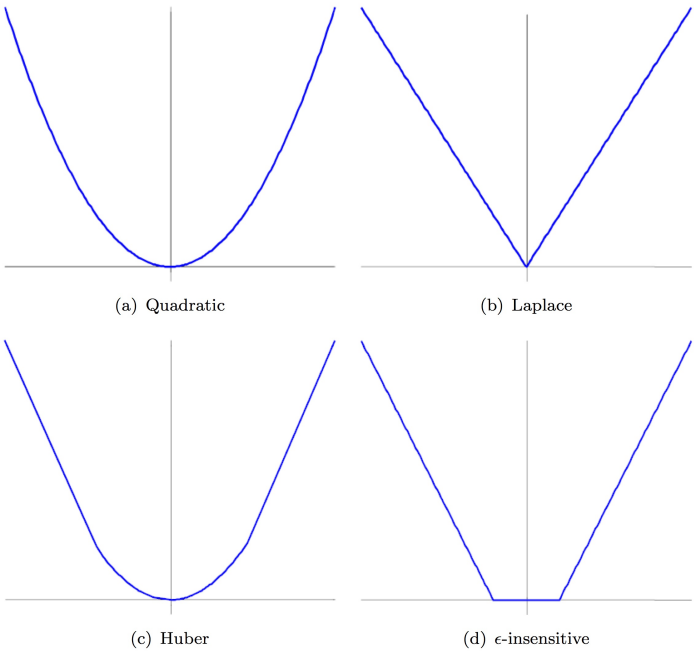

Figure 4: Types of loss functions, that can used in Support Vector Regression.

With different forms of loss function, the cost function takes different forms, although originally used and more popular is ε-sensitive loss function. Let's take a closer look at how Support Vector Regression (SVR) works in this case. At each point x_i we allow an error of ε. Everything above ε is captured in slack variables ξ_i^*, which are penalized in the objective function via a regularization parameter C, chosen apriori. And optimization problem can be written as equation (17), subject to equation (15) and equation (16). And as we have already seen before it is more efficient to solve this problem in dual formulation.
But how to choose ε-parameter? In some particular situations we can have knowledge of the model's noise level, so we choose the optimal ε, however it cannot be applied in general. To solve this issue, we could include ε in the optimization problem in order to minimize it. In this case our objective function will have form

$$\frac{1}{2}\|w\|^2 + C\left[\sum_{i=1}^{N}\xi_i + N\nu\varepsilon\right] \tag{24}$$

subject to equation (15) and equation (16). This formulation (also called as ν-regression) helps to control the number of support vectors directly. Essentially it allows the tube to adapt to the data automatically. It is also worth mentioning that the form of tube is remains the same, however, it also can be changed with parametric tube models with non-constant width. Here it is important to remember that ν is upper bound on the fractions of the errors and lower bound on the number of support vectors.

3 Forecasting application

Now let's use Support Vector Machine forecasting ability for predicting price of square foot of renting objects. As a benchmark we use linear regression. Also, as computing costs are called one of the disadvantages of SVM, we include them into research.

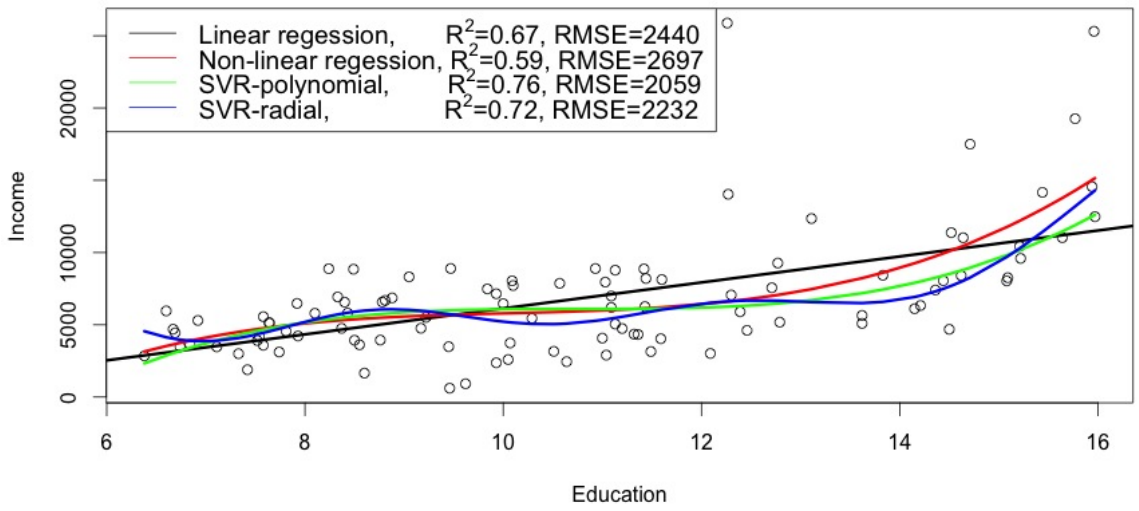

Figure 5: Comparison of Support Vector Regression with different kernels to linear and non-linear regressions by R-squared and root mean squared error on a dataset 'Prestige'.

3.1 Dataset

Our dataset consist of 4696 observations of 21 independent variables, which are characteristics of advertisement of renting object with two variables describing number of registered crimes in and amount of other renting object in the near distance. We split this dataset into eight logical data groups and predict price separately based on each group.

3.2 Evaluation

Initially we check the data for outliers, and exclude observations, if any of its values are significantly larger than average value. For prediction we construct one linear regression model and four SVM ones - with *C* and *nu* parameters with two kernels - radial and polynomial, as most popular for initial analysis - for each parameter. All models are trained on 70% of data and 30% are used for testing. As performance measures we use mean squared error and forecast skill. Forecast skill relates forecast accuracy of specific model to reference one, perfect forecast results in forecast skill value of *'1'*, while less skillful than reference has negative values. We also estimate computation costs, to do so we calculate time needed to build model and predict the price and for comparison we take average time value.

For our research we use the statistical software R. It contains two libraries for work with SVM - *e1071* and *kernlab*. First one is less complex - it has smaller amounts of kernel functions to choose, less parameters to optimise and misses some additional features. However, for our introduction analysis it fits better and all our calculations are made using this library.

To run Support Vector Machine we use function *'svm'*. It returns set of values, which can be used for later calculations. All parameters, like type of SVM, kernel functions, coefficients values, are set by default value and can be easily changed. Also *'svm'* is integrated with other popular functions, like *'plot'*, *'predict'* and *'tune'*, what makes it easy to use. *'Tune'* function is of particular

use for us, because it allows to find the best value of the model coefficients and parameters inside the given interval by means of grid search, and we use it in our evaluation to find best C and ν parameters in C- and ν-regressions accordingly.

3.3 Results

First finding that catches eyes, is that results of data group containing all variables is the worst in comparison to other data groups. This can be explained that different variables could have different causal effect on the rent price, thus, producing noise combined in one group.
Another interesting outcome is that in all data groups Support Vector Regression outperforms linear one, however, results are very tight, and this can be caused by original data - values of dependent variable are located very close to each other. Comparing results of the models with respect to parameters and kernels we see that they are also very tense, with slight advantage to combination of parameter C and polynomial kernel. The best predictions were given by data group 'Size', which seems to be logical, because there should be connection between rent price and income of rent object owners, meaning that owners of more expensive object have higher income and they can afford to have bigger (or, in other words, more expensive) advertisement of their offer and vice versa. Unexpected result gives group 'Environment' - one of the worst. This could mean that number of factors of real-world characteristics is not enough for good predictions or that we should search for additional characteristics, which could give better predictions.
Concerning computation costs results are more interesting. Linear regression is about 200 times faster in calculation than ν-regression. And C-regression 0,5 of second slower than using parameter ν. And if we will take a look at 'tuned' results we will see that although precision has increased, computation costs has also increased, but disproportionately high. If to talk about relatively small datasets with modern computational power of hardware - time is not crucial factor, but if we are going to deal with Big Data - computational costs are becoming more important and, sometimes, it could be better to sacrifice the accuracy of the results in order to save the time.
All results can be found in the Appendix.

4 Conclusion

Nowadays, data mining and machine learning are becoming valuable tools for companies. The more precise decision or accurate prediction can be given by the model, the better solution will company make or will more effectively react to the changes in the environment. That is why question of finding best algorithm is very important and Support Vector Machine is one the most popular ones.
Paper gives introduction to Support Vector Machine, it's main ideas, features, extensions and areas of application. It is very effective for classification of linearly separable data and also can be applied to non-linearly separable data. Soft margin and kernel tricks allow flexibility in choosing and tuning the model. One of main advantages of Support Vector Machine is minimizing expected generalization loss, what helps to produce good results even with small training sample. We also have tested Support Vector Machine against linear regression using dataset of rent price objects, and have compared average computational costs.

In future work we will take a look at another extension of Support Vector Machine - Relevance Vector Machine. In testing we will add more kernels (Gaussian and hyperbolic) and models (non-linear regression and neural networks), increase intervals for finding best C and ν parameters and tune Support Vector Regressions with respect to kernel coefficients as well. Also changing share of dataset used for training, can help us to find the optimal ratio. Regrouping variables, searching for new ones and running models separately with every single variable can give us new interesting results.

A References

[1] A. Karatzoglou, D. Meyer, K. Hornik. *Support Vector Machines in R*. In: Journal of Statistical Software, 2006.
[2] A. Smola, B. Scholkopf. *A tutorial on Support Vector Regression*. In: NeuroCOLT2 Technical Report Series, 1998.
[3] B. Scholkopf, A. Smola, R. Willaimson, P.Bartlett. *New support Vector Algorithms*. In: NeuroCOLT2 Technical Report Series, 1998.
[4] C. Cortes, V. Vapnik. *Support-Vector Networks*. In: Machine Learning, 20, 1995, pp. 273-297.
[5] C. Chang, C. Lin. *Training Support Vector Classifiers: Theory and Algorithms*. In: Neural Computation, 2001.
[6] D. Meyer, F. Leischa, K. Hornik. *The support vector machine under test*. In: Neurocomputing 55, pp. 169-186, 2003.
[7] S. Gunn. *Support Vector Machines for Classification and Regression*. In: ISIS Technical Report, 1998.
[8] C. Bishop. *Pattern recognition and machine learning*. New York, NY: Springer, 2006.
[9] S. Theodoridis, K. Koutroumbas. *Pattern recognition*. 3^{rd} Ed. Amsterdam; Heidelberg[u.a.]: Elsevier Academic Press, 2006.
[10] S. Russell, P. Norvig. *Artificial intelligence : a modern approach* 3^{rd} Ed. Boston and Munich: Pearson, 2010.

B Appendix

Type of regression	MSE	FS
Size		
Linear	0.01939	0.99263
C-Polynomial	**0.00697**	**0.99735**
C-Radial	0.01525	0.99420
ν-Polynomial	0.01454	0.99447
ν-Radial	0.01288	0.99510
Images		
Linear	0.07396	0.97187
C-Polynomial	0.03866	0.98530
C-Radial	0.07189	0.97266
ν-Polynomial	0.01756	0.99332
ν-Radial	**0.01671**	**0.99365**
Color		
Linear	0.13006	0.95054
C-Polynomial	**0.03957**	**0.98495**
C-Radial	0.07926	0.96986
ν-Polynomial	0.05256	0.98001
ν-Radial	0.05588	0.97875
Saturation		
Linear	0.04026	0.98469
C-Polynomial	**0.01494**	**0.99432**
C-Radial	0.04218	0.98396
ν-Polynomial	0.01622	0.99383
ν-Radial	0.02733	0.98960

Type of regression	MSE	FS
Pixels		
Linear	0.13679	0.94798
C-Polynomial	0.46791	0.82205
C-Radial	0.13603	0.94826
ν-Polynomial	0.35503	0.86498
ν-Radial	**0.11184**	**0.95747**
Tweets		
Linear	0.20413	0.92237
C-Polynomial	**0.03745**	**0.98576**
C-Radial	0.19163	0.92712
ν-Polynomial	0.04026	0.98469
ν-Radial	0.12806	0.95130
Environment		
Linear	0.39873	0.84835
C-Polynomial	0.06117	0.97673
C-Radial	0.38852	0.85224
ν-Polynomial	**0.09217**	**0.96494**
ν-Radial	0.22945	0.91273
Edge proportion		
Linear	0.02953	0.98877
C-Polynomial	**0.00178**	**0.99932**
C-Radial	0.03241	0.98767
ν-Polynomial	0.01469	0.99441
ν-Radial	0.00965	0.99633

Type of regression	MSE	FS
All parameteres		
Linear	0.50058	0.80962
C-Polynomial	2.23671	0.14933
C-Radial	0.46012	0.82501
ν-Polynomial	0.84198	0.67978
ν-Radial	**0.09169**	**0.96512**

Table 1: Results of evaluation. Best result in each group is highlited with bold font.

Type of regression	Linear	Tune	*C*	*ν*
Time	0.02	no	4.54	4.02
		yes	67.69	21.58

Table 2: Computational costs. Time is measured in seconds and calculated as average for all data groups. Since tuning is made only for parameters *C* and *ν* we take average time for all kernels.