# Machine Learning in R

Jakub Glinka

**Warsaw University**
**Department of Applied Statistics**
kubaglinka@mimuw.edu.pl

4 października 2009

- Supervised Learning (Support Vector Machines)
- Unsupervised Learning (Neural Networks)
- Semi-supervised Learning (co-training)
- Reinforcement Learning (Policy Estimation)
- Transduction (TSVM)

- make a diagnosis based on some clinical measurements;
- assign the ASCII code to digitalized images of handwritten characters;
- predict whether a client will pay back a loan to a bank;
- assess the price of house based on certain characteristics;
- estimates the costs of claims of insurees based on insurance data.

In the machine learning approach we assume that we have collected a sequence

$$D := ((x_1, y_1), ..., (x_n, y_n))$$

of input/output pairs, from known sets X, Y respectively, that are used to „learn" a *decision function* :

$$f_D : X \to Y$$

that is a *good approximation* of the possible response *y* to an arbitrary *x*.

Obviously, in order to find such function, it is necessary that the already collected data *D* have something in common with the new and unseen data. In the framework of machine learning theory, this is guaranted by assuming that both past and future pairs (x,y) are independently generated by *the same*, but of course *unknown*, probability **P** on *XxY*. Note that this is fundamental difference from parametric models, in which the relationship between the inputs *x* and the outputs *y* is assumed to follow some unknown function *f* from *known, finite-dimensional set of functions*.

**Def.** Let **P** be probability measure on XxY. For measurable function *f* we define **L-risk** as

$$R_{L,\mathbb{P}}(f) = \int_{X \times Y} L(x, y, f(x)) d\mathbb{P}(x, y)$$

where function L is non-negative and measurable. In case of empirical measure we get **empirical L-risk:**

$$R_{L,\mathbb{D}}(f) = \frac{1}{n} \sum_{i=1}^{n} L(x_i, y_i, f(x_i))$$

As primal optimization problem SVM introduces minimalisation of empirical counterpart of **regularized L-risk functional:**

$$R^{reg}_{L,\mathbb{P},\lambda}(f) := R_{L,\mathbb{P}}(f) + \lambda \|f\|^2_{\mathcal{H}},$$

where H is certain possibly *infinite-dimensional* Hilbert space of functions. One can show that without loss of generality

$$\inf_{f \in \mathcal{H}} R^{reg}_{L,\mathbb{P},\lambda}(f) = \inf_{f \in \lambda^{-\frac{1}{2}} B_{\mathcal{H}}} R^{reg}_{L,\mathbb{P},\lambda}(f),$$

and so, lambda is a trade-off between complexity and quality of solution.

This is the moment where *kernel trick* comes to a play. We call function

$$k : X \times X \rightarrow R$$

*a kernel* if there exists hilbert space *H* and mapping

$$\phi : X \rightarrow \mathcal{H}$$

that function *k* is inner product in that space, namely

$$\forall_{x,x' \in X} \quad k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}.$$

One can show that for every kernel there is a canonical mapping on certain *hilbert space H* given by

$$\phi(x) := k(\cdot, x), \quad x \in X,$$

such that, *k* has *reproducing property, namely*

$$\forall_{f \in \mathcal{H}} \quad \forall_{x \in X} \quad k(\cdot, x) \in \mathcal{H} \quad i \quad f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}}$$

and set of functions given below is dense in *H* and, most importantly, this is the set used by SVM for regularized empirical risk minimisation.

$$\mathcal{H}_{pre} := \{\sum_{i=1}^{n} \alpha_i k(\cdot, x_i) : \quad n \in N, \quad \alpha_1, ..., \alpha_n \in R, \quad x_1, ..., x_n \in X\}$$

Directly from the *reproducing property* of kernel it follows that

$$\forall_{f \in \mathcal{H}_{pre}} \quad \|f\|_{\mathcal{H}}^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) = \alpha K \alpha^{\top}$$

where K is semi-definite matrix, and so, if we consider convex loss function L, in order to find the decision function we have to solve *finite dimensional convex program :*

$$\inf_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(x_i, y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2 =$$

$$= \min_{\alpha \in R^n} \frac{1}{n} \sum_{i=1}^{n} L(x_i, y_i, e_i K \alpha^{\top}) + \lambda \alpha K \alpha^{\top}.$$

Examples of popular kernels:
*linear kernel*

$$k(x, x') = \langle x, x' \rangle \ , x, x' \in R^d$$

*polynomial kernel*

$$k(x, x') = (\langle x, x' \rangle + c)^m$$

*gaussian kernel*

$$k_{\gamma, R^d}(x, x') = \exp(-\frac{\|x - x'\|_2^2}{\gamma^2}), \quad x, x' \in R^d$$

As for the loss function L, there are two commonly used in practice. For classification tasks we have *hinge loss (or soft margin loss):*

$$L_{SM}(x, y, t) = \max\{0, 1 - yt\} \qquad x \in R^d, y \in \{-1, 1\}, t \in R.$$

which penalizes lineary for misclassification, and *epsilon insensitive loss* used mainly for regression problems:

$$L_\epsilon(x, y, t) = \max\{0, |y - t| - \epsilon\} \qquad x \in R^d, y \in \{-1, 1\}, t \in R.$$

If we consider simple binary classification task using soft margin loss we are faced with *primal problem* called *C-SVC*:

$$\min_{\alpha,\xi\in R^n} C\sum_{i=1}^{n}\xi_i + \frac{1}{2}\|w(\alpha)\|^2_{\mathcal{H}}$$

$$\xi_i \geqslant 1 - y_i w(\alpha)(x_i), \quad \xi_i \geqslant 0 \quad i = 1, ..., n$$

where

$$\forall_{\alpha\in R^n} \quad w(\alpha) = \sum_{i=1}^{n}\alpha_i\phi(x_i)$$

Points with ξ_i>0 are called *support vectors (SV).* One can show that point corresponding coeficient alpha is >0 iff this point is SV (sparsity of the solution). Usually one adds threshold *b* to find slightly different function

$$w_b(\alpha) = \sum_{i=1}^{n}\alpha_i\phi(x_i) + b$$

Because in some cases finding value of parameter C in C-SVC can be difficult there is nice modification of that previous algorithm called *nu-SVC* :

$$\min_{\alpha,\xi\in R^n,\rho,b\in R} \frac{1}{n}\sum_{i=1}^{n}\xi_i - \nu\rho + \frac{1}{2}\|w(\alpha)\|^2_{\mathcal{H}}$$

$$\xi_i \geqslant \rho - y_i(w(\alpha)(x_i) + b), \quad \xi_i \geqslant 0 \quad \rho \geqslant 0 \quad i = 1, ..., n$$

Lets assume that alghorithm nu-SVC gave rho>0, and denote

$$\frac{1}{n}\#\{i : y_i(w(\alpha)(x_i)+b) < \rho\}$$

as fraction of marginal errors. One can show that parameter nu is lower bound on that fraction (and upper bound on fraction of SV). It can be also shown that C-SVC with C equal to 1/rho produces the same solution.

If we consider regression task using epsilon-insensitive loss we are faced with *primal problem* called *epsilon-SVR*:

$$\min_{\alpha,\xi^+,\xi^- \in R^n} C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) + \frac{1}{2}\|w(\alpha)\|_{\mathcal{H}}^2$$

with respect to

$$\xi_i^+ \geqslant 0, \quad \xi_i^+ \geqslant y_i - w(\alpha)_b(x_i) - \epsilon \quad i = 1, ..., n$$

$$\xi_i^- \geqslant 0, \quad \xi_i^- \geqslant w(\alpha)_b(x_i) - y_i - \epsilon \quad i = 1, ..., n.$$

Note that thanks to this loss function we can achieve simillary to SVC alghorithms sparsity of the solution. Hovewer unlike then we have two parameters to adjust instead of one – C and epsilon. Epsilon can be considered as a priori set level of accuracy of solution but in most cases we want the solution to be as accurate as possible. To resolve this problem one makes epsilon part of optimisation problem.

This modification is called *nu-SVR,* with primal problem shown below:

$$\min_{\alpha,\xi^+,\xi^-,b,\epsilon\in R^n} C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^- + \nu\epsilon) + \frac{1}{2}\|w(\alpha)\|_{\mathcal{H}}^2$$

with respect to

$$\xi_i^+ \geqslant 0, \quad \xi_i^+ \geqslant y_i - w(\alpha)_b(x_i) - \epsilon \quad i = 1, ..., n$$

$$\xi_i^- \geqslant 0, \quad \xi_i^- \geqslant w(\alpha)_b(x_i) - y_i - \epsilon \quad i = 1, ..., n.$$

Lets assume that alghorithm nu-SVR ended with epsilon>0. One can show that nu is upper bound on fraction of errors, where as an error we denote points with distance of outputs more than epsilon from the decison function (in other words outside epsilon-tube of decision function). It is also lower bound on fraction of SVs.

The first implementation of SVM in R was introduced in the **e1071** package. The **svm()** function provides a rigid interface to **libsvm** along with visualization and parameter tuning methods. **Libsvm** is fast and easy to use implementation of the most popular SVM formulations: C-SVC, nu-SVC, epsilon and nu SVR. It includes the most common kernels (linear, polynomial, gaussian and sigmoid), only extensible by changing the C++ source code. For multi-class classification *one-against-one* voting scheme is used. Package basically provides a training function with standard and formula interfaces and a **predict()** method. Hyperparameter tuning is done using the **tune()** framework performing a grid search over specified parameter ranges. By default the error measure is computed using 10-fold cross validation on the given data.

C-SVC should be used in case of multiclassification problems. Below is sample code in R for prediction of class of age of abalone. We tune *hyperparameters* of gaussian kernel:

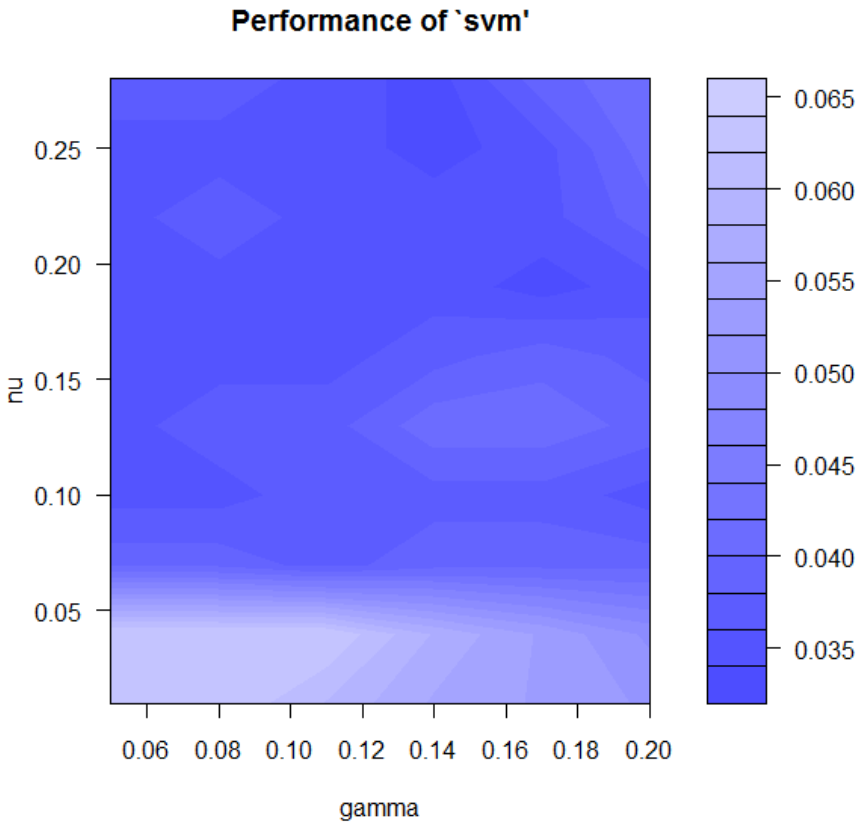
```
>tune.svm(age~.,data=train.data2,type="C-classification",kernel="radial",cost=seq(.5,2.5,.5),cachesize=100,cross=10,gamma=1/8)

  Parameter tuning of `svm':

  - sampling method: 10-fold cross validation

  - best parameters:
   gamma cost
   0.125  0.5

  - best performance: 0.28
```

We use best parameters to find a decision function and accuracy of classifier

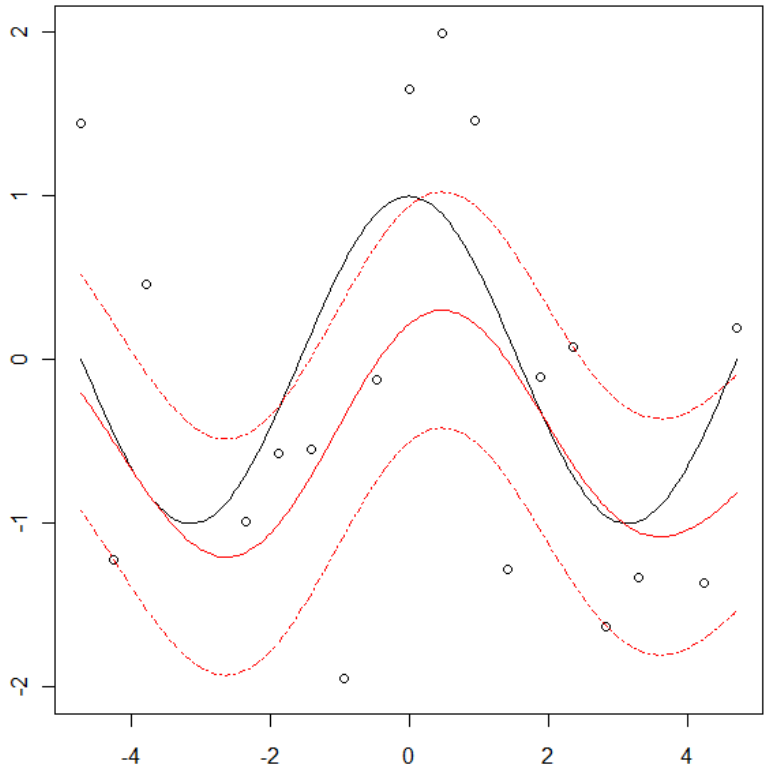
```
>model_radial<-svm(age~.,data=train.data2,type="C-classification",kernel="radial",cost=2,cross=10,gamma=1/8)
>mean(predict(model_radial,train.data2)==train.data2[,9])

  [1] 0.7266667
```

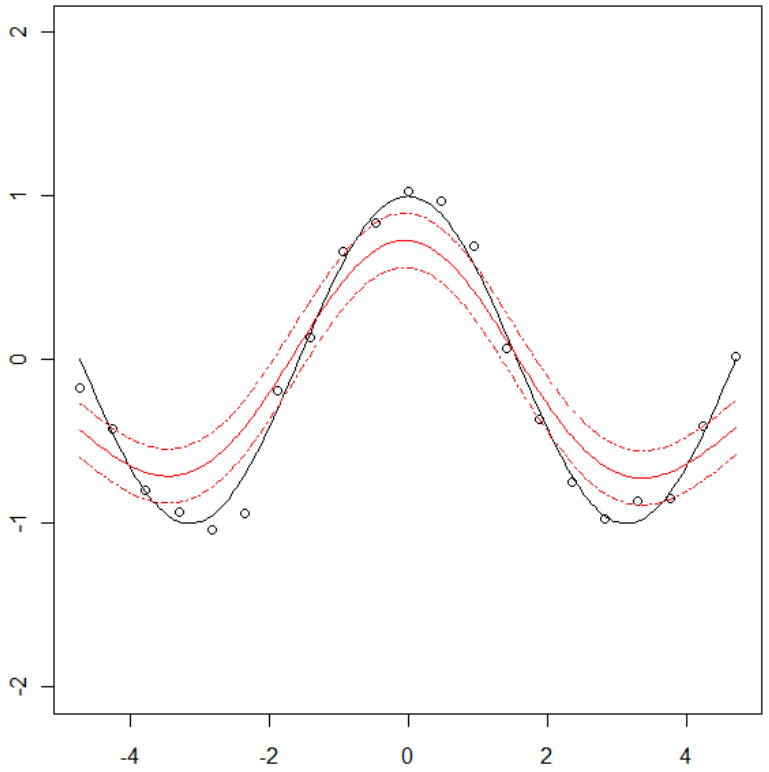
In most cases besides training accuracy we also asses accuracy on test set. Below is sample code of predicting malignant cancer type.

```
> model<-svm(type~.,data=train.data,type="nu-classification",kernel="radial",nu=.077,gamma=0.015)
> summary(model)

Call:
svm(formula = type ~ ., data = train.data, type = "nu-classification", kernel = "radial", nu = 0.077, gamma = 0.015)


Parameters:
   SVM-Type:  nu-classification
 SVM-Kernel:  radial
      gamma:  0.015
         nu:  0.077

Number of Support Vectors:  52

 ( 20 32 )


Number of Classes:  2

Levels:
 2 4

> mean(predict(model,train.data)==train.data$type)
[1] 0.9752577
> mean(predict(model,test.data)==test.data$type)
[1] 0.9626168
```



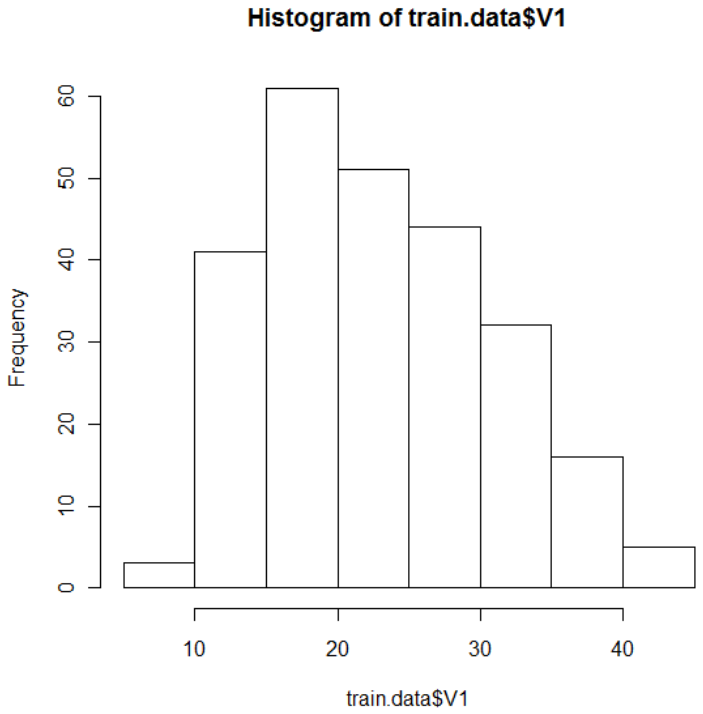
On the pictures below we see how width of the epsilon-tube changes with data noise. (nu=.6)



Below is R code for predicting fuel consumption on **Auto-mpg** dataset.

```
> model<-svm(V1~.,data=train.data,type="nu-regression",kernel="radial",nu=.6,gamma=1/8,cost=1)
> summary(model)

Call:
svm(formula = V1 ~ ., data = train.data, type = "nu-regression", kernel = "radial", nu = 0.6, gamma = 1/8, cost = 1)


Parameters:
   SVM-Type:  nu-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.125
         nu:  0.6

Number of Support Vectors:  213

> mean(abs(predict(model,train.data)-train.data$V1)/train.data$V1)
[1] 0.04426202
> mean(abs(predict(model,train.data)-train.data$V1))
[1] 1.089199
> mean(abs(predict(model,test.data)-test.data$V1)/test.data$V1)
[1] 0.09294827
> mean(abs(predict(model,test.data)-test.data$V1))
[1] 2.231173
```



| | | **ksvm()** (kernlab) | **svmlight()** (klaR) | **svmpath()** (svmpath) |
|---|---|---|---|---|
| Formulations | | C-SVC, nu-SVC, C-BSVC, spoc-SVC, one-SVC, eps-SVR, nu-SVR, eps-BSVR | C-SVC, eps-SVR | binary C-SVC |
| Kernels | | Gaussian, polynomial, linear, sigmoid, Laplace, Bessel, Anova, Spline | Gaussian, polynomial, linear, sigmoid | Gaussian, polynomial, linear |
| Model Selection | | Hyperparameter estimation for Gaussian kernels | NA | regularization path for cost parameter |
| Extensibility | | custom kernel functions | NA | custom kernel functions |

[1] Bernhard Schölkopf, A. J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge, MA,2002. MIT Press.

[2] Vladimir N. Vapnik. Statstical Learning Theory. 1998. Wiley and Sons.

[3] Felipe Cucker, Ding Xuan Zhou. Learning Theory: An Approximation Theory Viewpoint. 2007. Cambridge University Press.

[4] Ingo Steinwart, Andreas Christmann. Support Vector Machines. Information Science and Statistics. 2008. Springer.

[5] Statistical Learning in R. Journal of Statistical Software. 2007.

[6] Jacek Jakubowski, Rafał Sztencel. Wstęp do teorii prawdopodobieństwa. Wydanie III. Warszawa 2004. SCRIPT.

[7] Walter Rudin. Analiza Funkcjonalna. Warszawa 2009. PWN

[8] Sam Waugh. Extending and benchmarking Cascade-Correlation. Phd Thesis, Computer Science Department. 1995 University of Tasmania.

[9] David Clark, Zoltan Schreter, Anthony Adams. A Quantitative Comparison of Dystal and Backpropagation. Australian Conference on Neural Networks (ACNN'96).

[10] O. L. Mangasarian, W. H. Wolberg. Cancer diagnosis via linear programming. SIAM News, Volume 23, Number 5, September 1990, pp 1 - 18.

[11] Cios, K.J., Wedding, D.K. Liu. CLIP3: cover learning using integer programming. Kybernetes, 26:4-5, pp 513-536, 1997