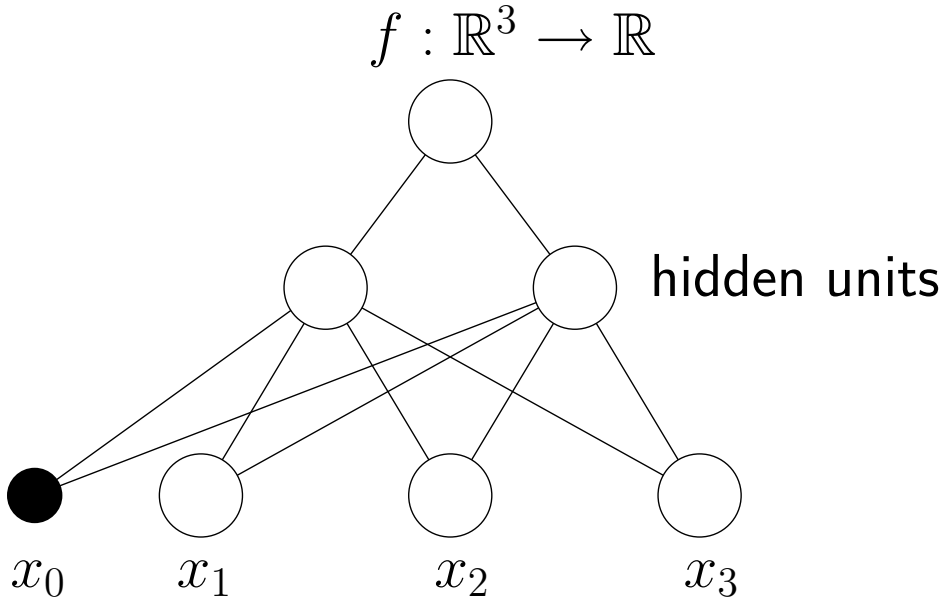
Regression Estimation

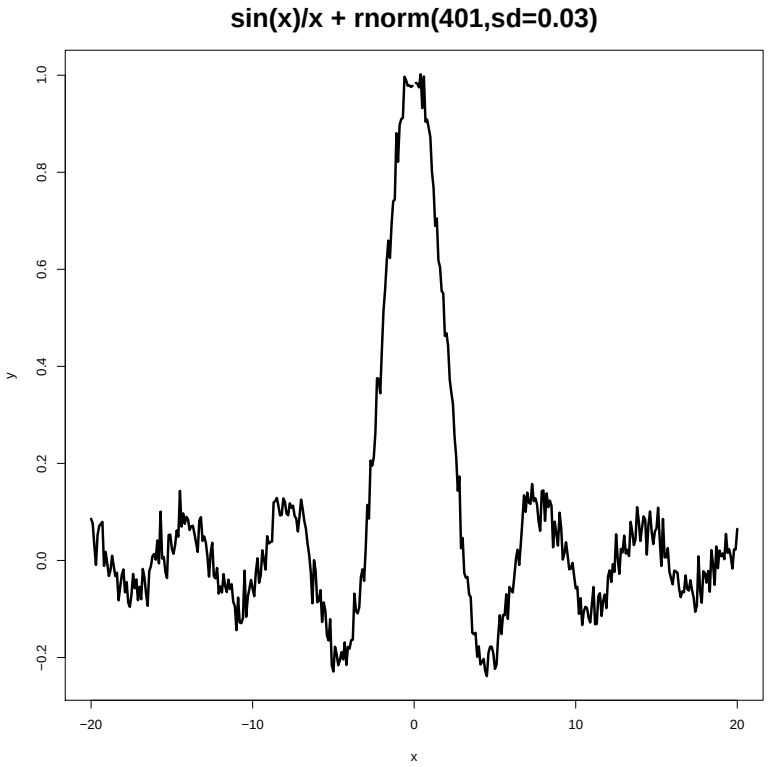
In the problem domain of supervised learning we have considered targets that are values from $\{-1, +1\}$. We now focus on data of input-output pairs of the following form:

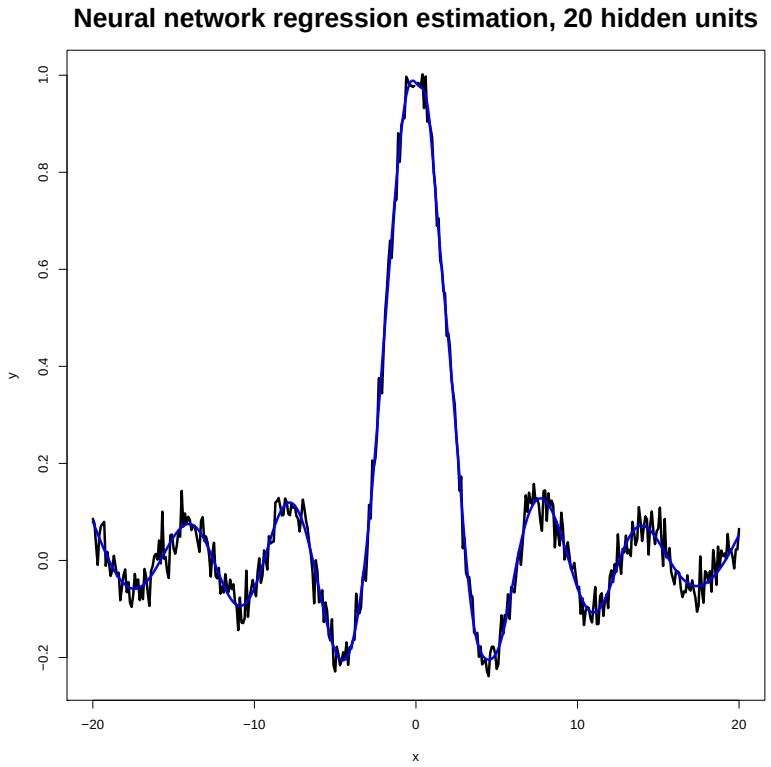
$$(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$$

Given data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$ drawn from $P(\mathbf{x}, y)$, choose a function $f : \mathbb{R}^d \to \mathbb{R}$ such that the error, averaged over P, is minimized.



Regression Estimation Example





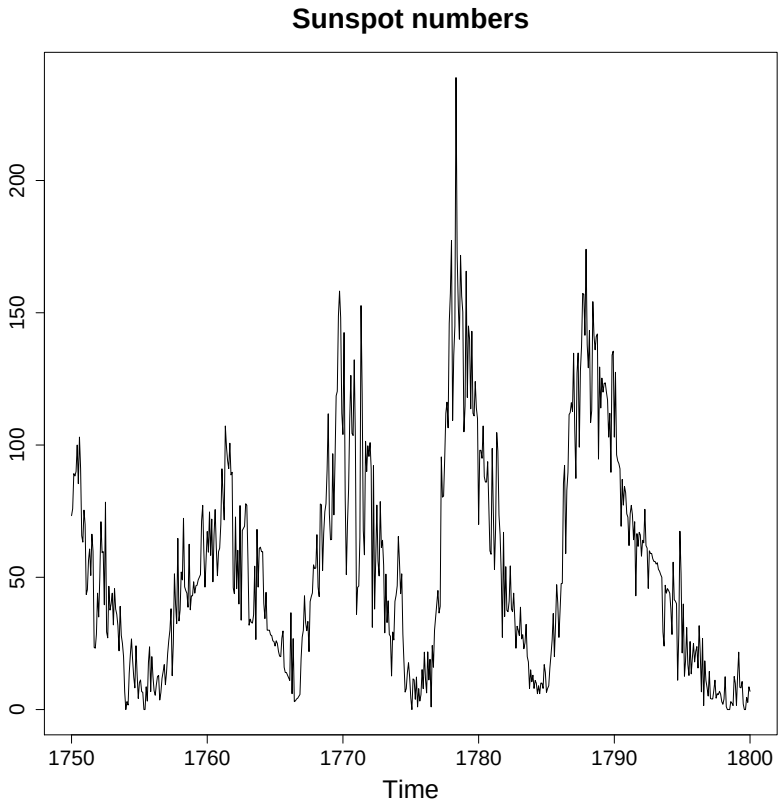

```
library(nnet);
x <- seq(-20,20,0.1);
y <- sin(x)/x + rnorm(401,sd=0.03);
# train neural network
reg.net<-nnet(y~x,linout=TRUE,size=20,maxit=500);
plot(x,y,type="l",lwd=3);
lines(x,predict(reg.net,x),col="blue",lwd=3);
```

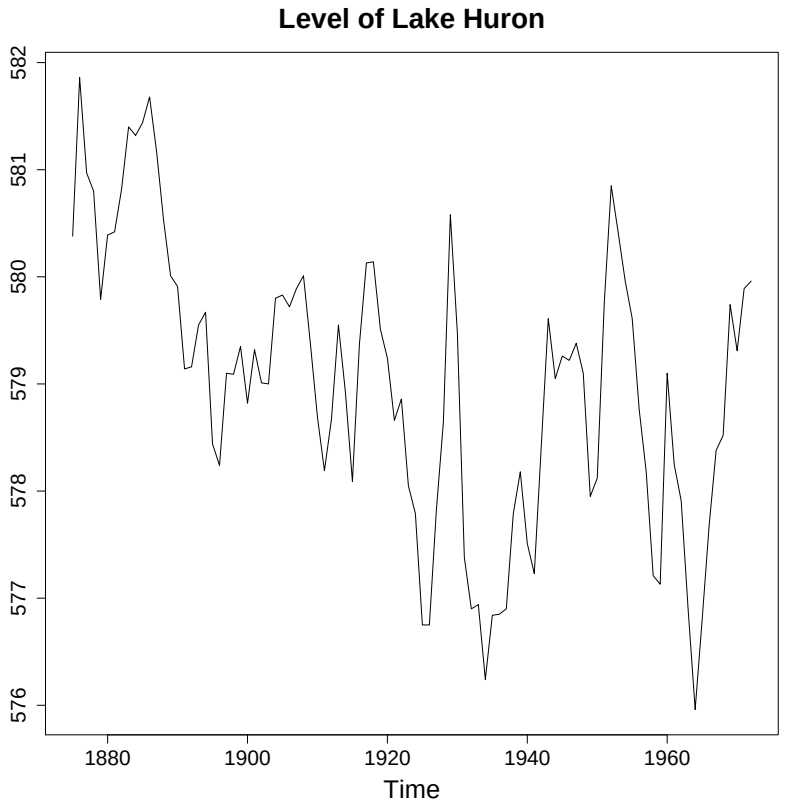
Neural Networks for Time Series Prediction

Data consists of scalars (or vectors) which depend on time t

$$\{x[t_0], x[t_1], \ldots, x[t_{i-1}], x[t_i], x[t_{i+1}], \ldots\}$$

Examples of time series





Predicting the Future

We have time series data and want to estimate x at some future time $\widehat{x}[t+p] = f(x[t], x[t-1], \ldots)$ where p is called the horizon of prediction.

For $p = 1$ one predicts just one time sample in the future.

More precisely, the network takes as input a state vector which is defined as

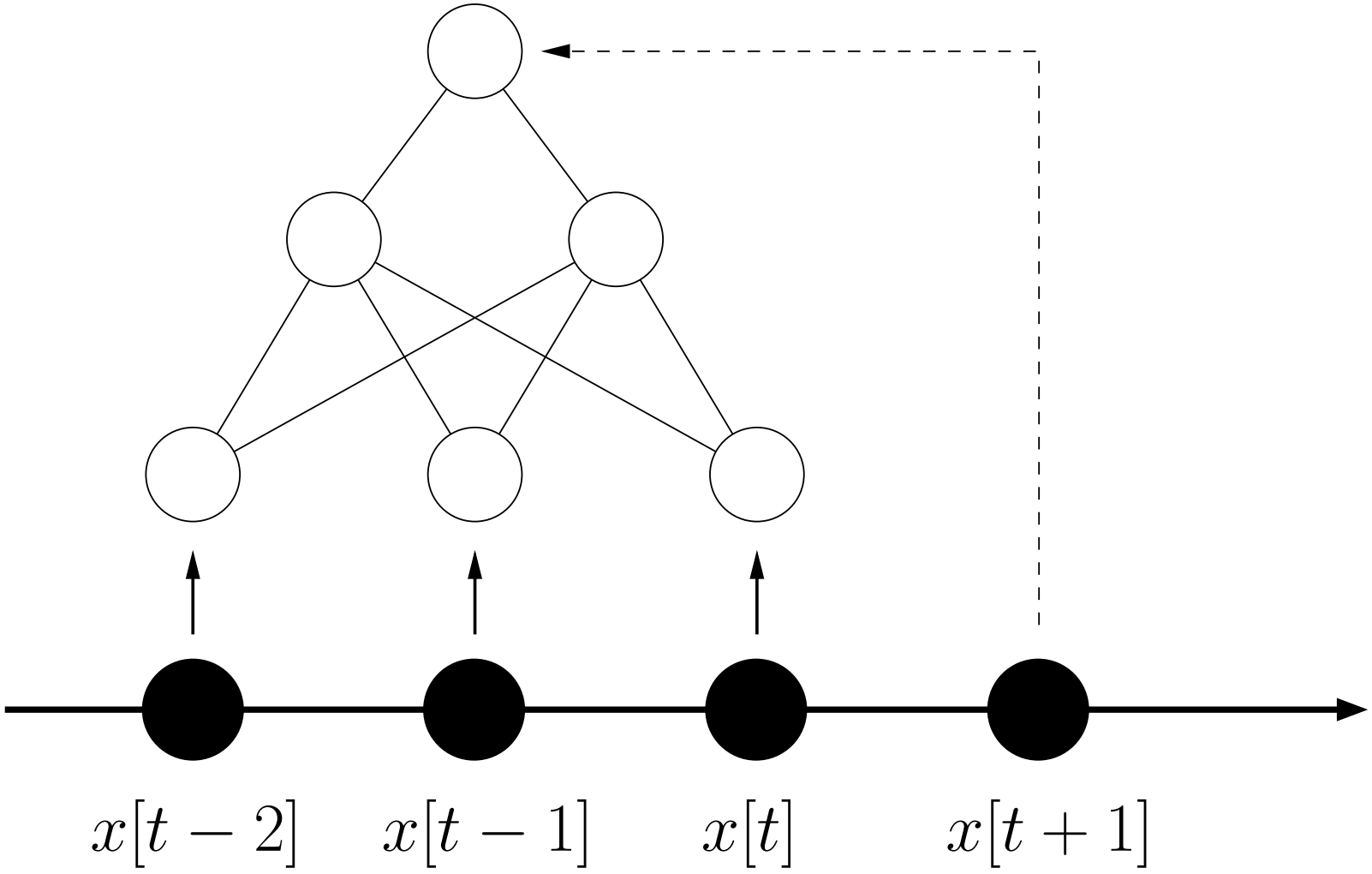
$$\mathbf{x}_t = (x[t], x[t-\tau], \ldots, x[t-(d-1)\tau])$$

where τ is the *time-delay* and d the *embedding* dimension, and predicts

$$\widehat{x}[t+p] = f(\mathbf{x}_t)$$

Feed-Forward NN for Time Series Prediction

$$f(\mathbf{x}_t) = \widehat{x}[t+1]$$



Horizon of Prediction

In most time series problems one does want predict $x[t+p]\,(p>1)$ rather than just the next value $x[t+1]$.

- *Iterated prediction:* feeding the previously predicted values $\widehat{x}[t+1], \widehat{x}[t+2], \ldots, \widehat{x}[t+p-1]$ as inputs of the network to predict $x[t+p]$.
- *Direct prediction:* train the network to learn directly $x[t+p]$ using $x[t], x[t-1], \ldots, x[t-d-1]$ as inputs.

Direct prediction can only be used when horizon p is small.

- Another alternative: neural network with p output units.

Support Vector Regression

Basic idea: map the data $\mathbf{x}$ into a high-dimensional feature space $\mathcal{F}$ via a nonlinear mapping Φ, and do *linear* regression in this space.

$$f(\mathbf{x}) = (\mathbf{w} \cdot \Phi(\mathbf{x})) + b \text{ with } \Phi : \mathbb{R}^d \rightarrow \mathcal{F}, \mathbf{w} \in \mathcal{F}.$$

Linear regression in a high dimensional feature space corresponds to *nonlinear* regression in the low dimensional space $\mathbb{R}^d$.

Vapnik's ϵ-insensitive loss function:

$$|y - f(\mathbf{x})|_\epsilon := \max\{0, |y - f(\mathbf{x})| - \epsilon\}$$

Find function $f(\mathbf{x})$ that has at most ϵ deviation from all the targets y_i

Support Vector Regression (cont.)

Estimate linear regression $f(\mathbf{x}) = (\mathbf{w} \cdot \Phi(\mathbf{x})) + b$ leads to the problem of minimizing the term

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} |y_i - f(\mathbf{x}_i)|_\epsilon$$

In the soft margin case one needs two types of slack variables $(\boldsymbol{\xi}, \boldsymbol{\xi}^*)$ for the two cases $f(\mathbf{x}_i) - y_i > \epsilon$ and $y_i - f(\mathbf{x}_i) > \epsilon$.

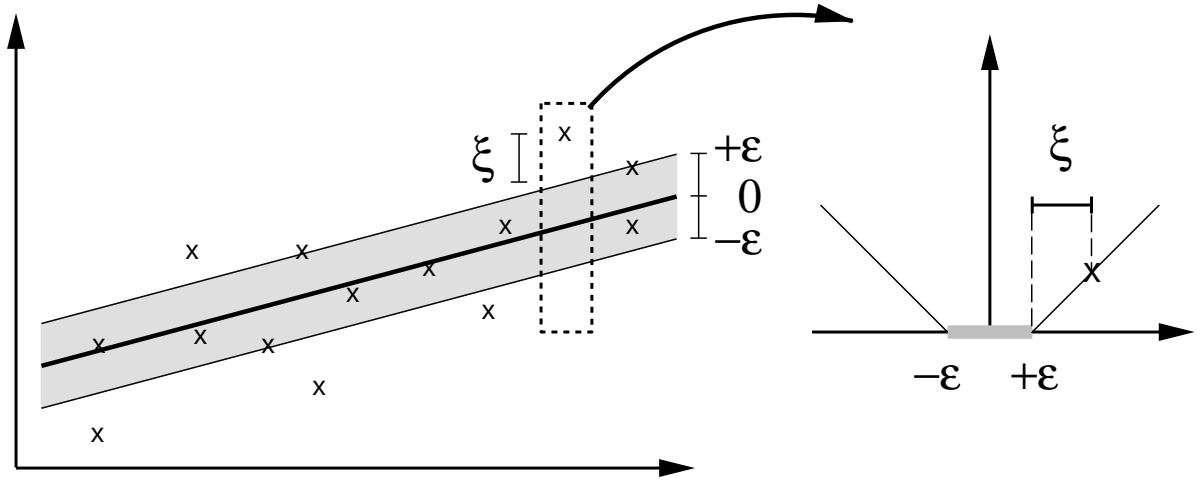


Figure is taken from Schölkopf's and Smola's book (Learning with Kernels)

Support Vector Regression (cont.)

Optimization problem is given by:

$$\text{minimize} \qquad \frac{1}{2}\|\mathbf{w}\|^2 + C \cdot \sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\begin{aligned} \text{subject to} \qquad f(\mathbf{x}_i) - y_i &\leq \epsilon + \xi_i \\ y_i - f(\mathbf{x}_i) &\leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0 \qquad \text{for all } i = 1, \ldots, n \end{aligned}$$

Support Vector Regression (cont.)

Introducing Lagrange multipliers $\boldsymbol{\alpha}, \boldsymbol{\alpha}^*$ (dual form):

$$\text{maximize} \quad -\epsilon \sum_{i=1}^{n}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)y_i$$

$$-\frac{1}{2}\sum_{i,j}^{n}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)k(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to} \quad 0 \leq \alpha_i, \alpha_i^* \leq C \text{ for all } i = 1, \ldots, n \text{ and}$$

$$\sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0$$

Regression estimate takes the form

$$f(\mathbf{x}) = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)k(\mathbf{x}_i, \mathbf{x}) + b$$

Support Vector Regression (cont.)

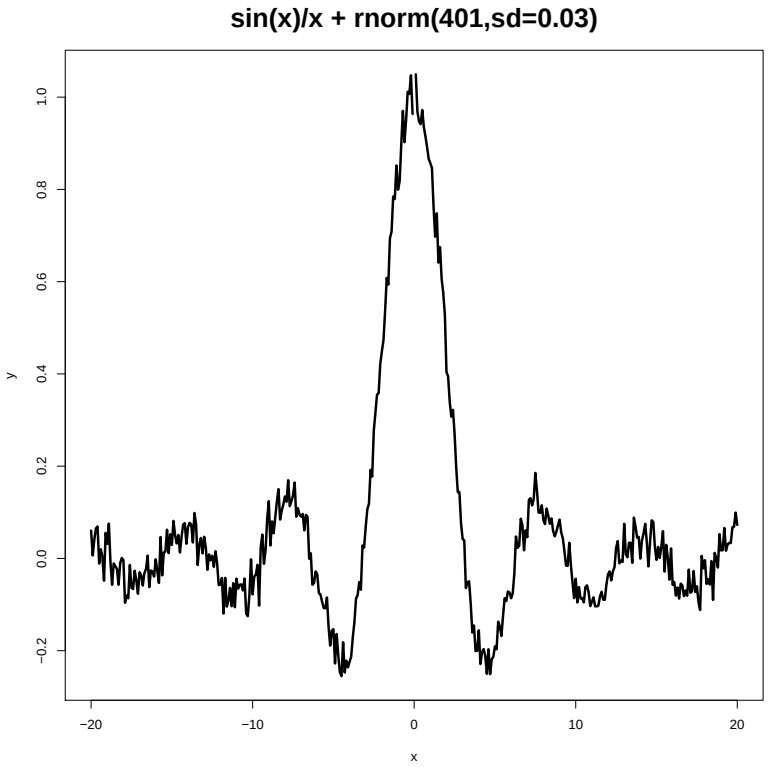
Offset b can be computed by exploiting Karush-Kuhn-Tucker conditions: $f(\mathbf{x}_i) - y_i \leq \epsilon + \xi_i$ becomes an equality with $\xi_i = 0$ if $0 < \alpha_i < C$ and $y_i - f(\mathbf{x}_i) \leq \epsilon + \xi_i^*$ becomes an equality with $\xi_i^* = 0$ if $0 < \alpha_i^* < C$ that is:

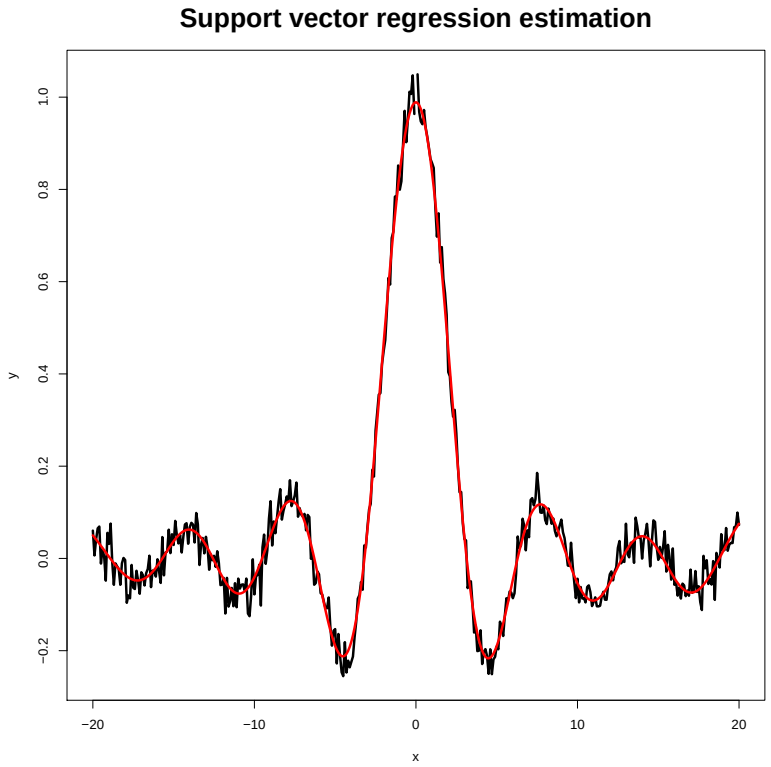
$$\begin{aligned} \alpha_i(\epsilon + \xi_i - y_i + (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) &= 0 \\ \alpha_i^*(\epsilon + \xi_i^* + y_i - (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) - b) &= 0 \end{aligned}$$

and leads to solution

$$\begin{aligned} b &= y_i - (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) - \epsilon \quad \text{for } \alpha_i \in (0, C) \\ b &= y_i - (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + \epsilon \quad \text{for } \alpha_i^* \in (0, C) \end{aligned}$$

Support Vector Regression Example






```
library(kernlab);
x <- seq(-20,20,0.1);
y <- sin(x)/x + rnorm(401,sd=0.03);
# train SVM
reg.svm <- ksvm(x,y,epsilon=0.01,kpar=list(sigma=16),cross=3);
plot(x,y,type="l",lwd=3);
lines(x,predict(reg.svm,x),col="blue",lwd=3);
```