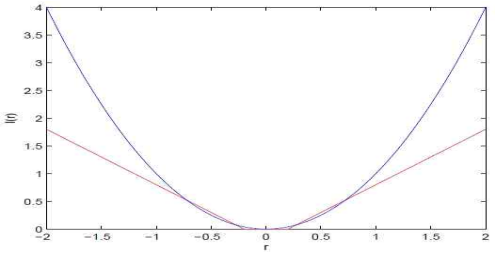
SVR(Support Vcetor Machine for Regression)

주어진 훈련자료 $(\boldsymbol{x}_i, y_i)_{i=1}^n$를 이용하여 비선형함수인 $f(\boldsymbol{x}_t)$를 예측.

다음과 같은 목적함수(objective function)를 최소화하는 관련 모수를 구해서 $f(\boldsymbol{x}_t)$를 예측.

$$\min_f \quad L_0 = \frac{1}{2}\boldsymbol{\omega}'\boldsymbol{\omega} + C\sum_{i=1}^{n}\ell_e(y_i - f(\boldsymbol{x}_i)),\ f(\boldsymbol{x}_i) = \boldsymbol{\omega}'\phi(\boldsymbol{x}_i) + b$$

여기서 $\ell_e(r) = (|r| - e)I(|r| > e) + 0I(|r| \le e)$ (e-insensitive loss function), $I(\cdot)$ 는 지시함수(indicator function), C>0 는 벌칙모수(penalty parameter).



빨간색: e-insensitive loss function (e=0.2)

파란색: quadratic loss function (제곱 손실함수)~ 일반적으로 추정에 많이 쓰이는 손실함수

주어진 벌칙상수와 커널 모수, e값, kernel trick, Lagrange method, KKT 조건을 적용하면 훈련자료 $(\boldsymbol{x}_i, y_i)_{i=1}^n$를 이용하여 Lagrange 배수 $(\alpha_i, \alpha_i^*)_{i=1}^n$ 와 bias b 를 구하기.

(Lagrange 배수 (α_i, α_i^*)에서 항상 $\alpha_i \ge 0$ or $\alpha_i^* \ge 0$, and $\alpha_i \times \alpha_i^* = 0$ ~ α_i, α_i^* 둘 중 하나 이상은 0)

주어진 $\boldsymbol{x}_t$에 대하여,

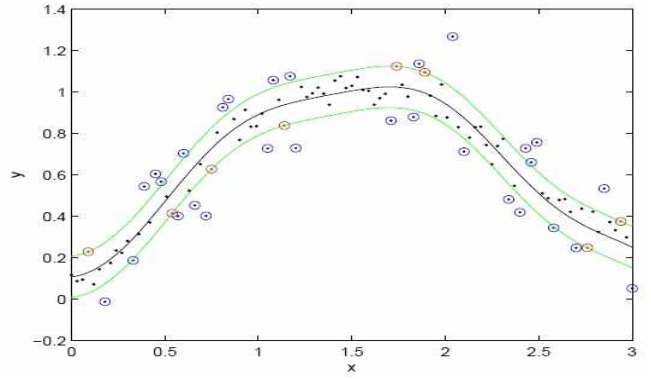
$\hat{f}(\boldsymbol{x}_t) = \sum_{i=1}^{n} K(\boldsymbol{x}_t, \boldsymbol{x}_i)(\alpha_i - \alpha_i^*) + b$ 가 구해짐.

여기서, $K(\boldsymbol{x}_t, \boldsymbol{x}_i)$: $\boldsymbol{x}_t, \boldsymbol{x}_i$의 커널함수, RBF or linear 등

RBF: $K(\boldsymbol{x}_t, \boldsymbol{x}_i) = \exp(-\frac{1}{\sigma^2}(\boldsymbol{x}_t - \boldsymbol{x}_i)'(\boldsymbol{x}_t - \boldsymbol{x}_i))$, $\sigma^2 > 0$: 커널모수, for nonlinear regression

linear: $K(\boldsymbol{x}_t, \boldsymbol{x}_i) = \boldsymbol{x}_t'\boldsymbol{x}_i$, for linear regression

예제) $\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} K(\boldsymbol{x}, \boldsymbol{x}_i)(\alpha_i - \alpha_i^*) + b$, K=RBF 커널



• : $\alpha_i = \alpha_i^* = 0$에 대응되는 data point

⊙(빨간색): $0 < \alpha_i < C, \alpha_i^* = 0$ 혹은 $\alpha_i = 0, 0 < \alpha_i^* < C$ 에 대응되는 data point

⊙(파란색): $\alpha_i = C, \alpha_i^* = 0$ 혹은 $\alpha_i = 0, \alpha_i^* = C$ 에 대응되는 data point

자료 중 $\alpha_i = \alpha_i^* = 0$에 대응되는 data point가 많음 & $\alpha_i = \alpha_i^* = 0$에 대응되는 data point가 추정에 사용되지 않음 ~ 추정에 사용되는 자료가 많지 않음 ~ 일반화성능(예측력)이 좋음.

<test_SVR_NNr_eg.py>

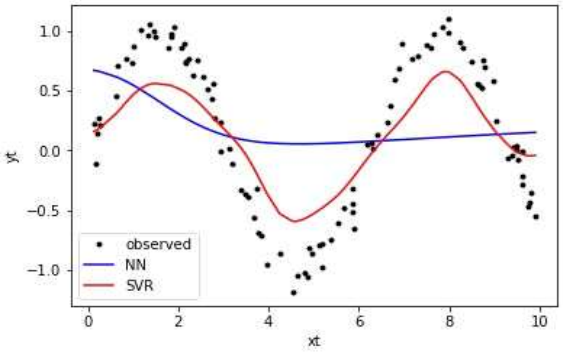
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 자료만들기
#x~ U(0.5), e~N(0,0.01), y=sim(x)+e
n=200; nt=100
np.random.seed(0)
x=10*np.random.uniform(0,1,n); xt=10*np.random.uniform(0,1,nt)
x=np.sort(x); xt=np.sort(xt)
e=0.1*np.random.normal(0,1,n); et=0.1*np.random.normal(0,1,nt)
y=np.sin(x)+e; yt=np.sin(xt)+et
x=x.reshape(n,1)
xt=xt.reshape(nt,1)

#NN reg
from sklearn.neural_network import MLPRegressor
# 은닉층수=1, 은닉층의 노드수=3, 활성함수=tanh
model1 = MLPRegressor(hidden_layer_sizes=(3), activation='tanh') .fit(x, y)
# 시험자료의 y예측값
ythN=model1.predict(xt)
# 시험자료의 y예측값의 mse
print('MSE NN=',np.mean((yt-ythN)**2))

# SVR
from sklearn.svm import SVR
# RBF kernel, C=100, sig2=0.5
model2 = SVR(kernel='rbf',C=100, gamma=1/0.5, epsilon=0.5).fit(x, y)
# 시험자료의 y예측값
yth=model2.predict(xt)
# 시험자료의 y예측값의 mse
print('MSE SVR=',np.mean((yt-yth)**2))
```

```
plt.figure(1)
plt.plot(xt,yt,'k.',label='observed')
plt.plot(xt,ythN,'b-',label='NN')
plt.plot(xt,yth,'r-',label='SVR')
plt.legend()
plt.xlabel('xt');plt.ylabel('yt')
```



* 예제자료의 예측성능은 SVR이 더 좋음.