Machine Learning. Homework 4

Skoltech 2015

Deadline: **8 December 2015, 16:00**.

1 Support Vector Regression

Consider a regression problem. We are given training data $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbf{R}^d$ is the feature vector of the ith training example and $y_i \in \mathbf{R}$ is the response. The task is to predict the response $y_{\text{new}} \in \mathbf{R}$ for a new (previously unseen) example with the feature vector $x_{\text{new}} \in \mathbf{R}^d$.

Support Vector Regression (SVR) is a regression model which predicts the response with a linear function:

$$\hat{y}(x) := w^\top \phi(x),$$

where $\phi : \mathbf{R}^d \to \mathcal{F}$ is some feature transformation (change of basis) and $w \in \mathcal{F}$ is the parameters of the model which are adjusted during the training process.

For training SVR, we solve the following ℓ_2-regularized empirical risk minimization problem:

$$\min_w \quad \frac{1}{n}\sum_{i=1}^n L(\hat{y}(x_i), y_i) + \frac{1}{2C}\|w\|_2^2, \tag{1}$$

where $L : \mathbf{R} \times \mathbf{R} \to \mathbf{R}_+$ is the Vapnik ε-insensitive loss, given by

$$L(\hat{y}, y) := \max\{0, |\hat{y} - y| - \varepsilon\}.$$

The values $\varepsilon > 0$ and $C > 0$ are the hyperparameters of SVR and are adjusted with cross-validation.

Optimization problem (1) is convex, unconstrained but non-smooth. We can make it smooth by introducing additional variables $\xi_i := \max\{0, |w^\top \phi(x_i) - y_i| - \varepsilon\}$. Then problem (1) is equivalent to

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{n}\sum_{i=1}^n \xi_i + \frac{1}{2C}\|w\|_2^2, \\ \text{s. t.} \quad & \xi_i \geq |w^\top \phi(x_i) - y_i| - \varepsilon, \qquad i = 1, \ldots, n, \\ & \xi_i \geq 0, \qquad\qquad\qquad\quad i = 1, \ldots, n. \end{aligned}$$

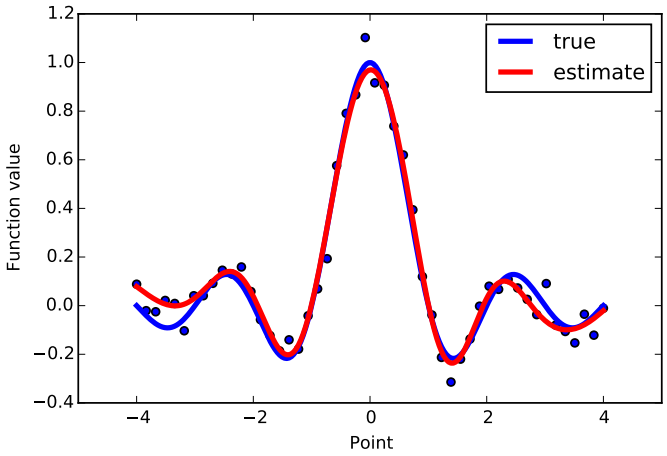

Figure 1: The result of SVR on the function $f(x) = \text{sinc}(x)$.

This problem is still non-smooth due to the presence of the absolute value function $|\cdot|$ in the second constraint. Rewriting this non-smooth constraint as two smooth constraints, we arrive at the smooth problem

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{n}\sum_{i=1}^{n}\xi_i + \frac{1}{2C}\|w\|_2^2, \\ \text{s.t.} \quad & w^\top\phi(x_i) - y_i \le \xi_i + \varepsilon, && i = 1,\dots,n, \\ & w^\top\phi(x_i) - y_i \ge -\xi_i - \varepsilon, && i = 1,\dots,n, \\ & \xi_i \ge 0, && i = 1,\dots,n. \end{aligned} \tag{2}$$

This is a convex Quadratic Programming (QP) problem which can be efficiently solved if the dimensionality of the new feature space $\mathcal{F}$ is not too big and the feature transformation ϕ can be computed explicitly.

If either of these does not hold, we use the kernel trick. For this we need to pass to the dual problem:

$$\begin{aligned} \max_{\lambda,\mu} \quad & -\frac{1}{2}(\mu-\lambda)^\top K(\mu-\lambda) + y^\top(\mu-\lambda) - \varepsilon e^\top(\mu+\lambda) \\ \text{s.t.} \quad & \lambda \succeq 0, \quad \mu \succeq 0, \\ & \lambda + \mu \preceq \frac{C}{n}e, \end{aligned} \tag{3}$$

where $K \in \mathbf{S}^n_{++}$ is the kernel matrix with elements $k_{ij} := \phi(x_i)^\top\phi(x_j)$, $y := (y_1,\dots,y_n) \in \mathbf{R}^n$ is the vector of outputs, and $e := (1,\dots,1) \in \mathbf{R}^n$ is the vector of ones. Since problem (2) is convex with linear constraints, strong duality holds, and so problem (2) is equivalent to its dual (3). This dual problem is again QP and can be efficiently solved. Once an optimal solution (λ^*, μ^*) is found, we can recover an optimal primal solution as follows:

$$w^* = \Phi(X)^\top(\mu - \lambda),$$

where $\Phi(X)$ is the matrix with rows $\phi(x_1),\dots,\phi(x_n)$.

Since everything depends on ϕ only through the kernel (dot products of $\phi(x_i)$ and $\phi(x_j)$), instead of explicitly specifying the feature transformation ϕ, we can specify the kernel function $k : \mathbf{R}^d \times \mathbf{R}^d \to \mathbf{R}$. Here are the most popular choices of kernel functions:

1. Linear (trivial) kernel:
$$k(x, x') = x^\top x'.$$
2. Polynomial kernel:
$$k(x, x') = \left(1 + x^\top x'\right)^p,$$
where $p \in \mathbf{N}$ is a hyperparameter called *degree*.
3. Radial Basis Function (RBF) kernel:
$$k(x, x') = \exp\left(-\gamma\left\|x - x'\right\|_2^2\right),$$
where $\gamma > 0$ is a hyperparameter called *bandwidth*.

2 Task

1. Rewrite problem (2) in the vector form, i.e. get rid of the sum and rewrite the constraints in the form $Au \preceq v$, where A is some matrix, and u and v are some vectors.
2. Derive the dual problem to (2). You should get problem (3). What is the dimensionality of this problem?
3. Suppose you have an optimal solution (λ^*, μ^*) of (3). Derive the formula for the prediction $\hat{y}(x_{\text{new}})$ which will not require to compute the feature transformation ϕ explicitly (i.e. it will contain only the corresponding kernel).

4. Show how problem (3) can be rewritten as a convex QP in the standard form:

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top P x + q^\top x, \\ \text{s.t.} \quad & Gx \preceq h. \end{aligned}$$

   Write the formulas which show what x, P, q, G and h are in the case of problem (3). Why will the matrix P be positive semi-definite?

5. Implement SVR using the formulas you derived. Your implementation must support three types of kernels: linear, polynomial and RBF.

6. Generate some synthetic one-dimensional training data:

$$y_i = f(x_i) + \zeta_i, \qquad i = 1, \ldots, n,$$

   where $f : \mathbf{R} \to \mathbf{R}$ is some deterministic function (any function you like) and $\zeta_i \sim \mathcal{N}(0, \sigma^2)$ is random noise with some variance $\sigma^2 > 0$. Fit your SVR on these data (with default hyperparameters). Plot its prediction. Make a picture similar to picture 1.

7. Try to vary the regularization parameter C. What changes? What about ε?

8. Find the best values of parameters C and ε using cross-validation. Has the prediction improved?

9. Investigate how the choice of a kernel affects the prediction (visually). What is the influence of the degree p of the polynomial kernel and the bandwidth γ of the RBF kernel?

10. For a given kernel (say, RBF), investigate how the accuracy and training time change in dependency of the number n of training examples. Do the same for the `SVR` class from sklearn. How do its results differ from your results?

11. Choose any simple real-world regression problem (e.g., use the Boston data set by calling `load_boston` from `sklearn.datasets`). Split these data into train and test. Compare the accuracy of your implementation of SVR and Kernel Ridge Regression (KRR) from sklearn on this data set. For choosing hyperparameters, you can use grid search.

3 Recommendations

1. For solving QP, you can use the CVXOPT package. See the corresponding example: `http://cvxopt.org/examples/tutorial/qp.html`.

2. You can easily compute the kernel matrix for linear, polynomial and RBF kernels using the functions `linear_kernel`, `polynomial_kernel` and `rbf_kernel` from the `sklearn.metrics.pairwise` module.

3. You can use many useful tools from the sklearn package such as grid search if you implement your SVR as a sklearn regressor. For this you need to derive your class from the `BaseEstimator`, `RegressorMixin` classes from the `sklearn.base` module and implement the methods `fit` and `predict`. Here is example code (file `mysvm.py`):

```
import numpy as np
from sklearn.base import BaseEstimator, RegressorMixin
from sklearn.metrics.pairwise import rbf_kernel, linear_kernel

class mySVR(BaseEstimator, RegressorMixin):
    def __init__(self, kernel='rbf', C=1.0, epsilon=0.1, gamma=1.0, p=1):
        self.kernel = kernel
        self.C = C
        self.epsilon = epsilon
        self.gamma = gamma
        self.p = p
```

```
    def fit(self, X, y):
        # Here is your code for training
        self.w = np.zeros(X.shape[1])

    def predict(self, X_new):
        # Here is your code for prediction
        y_new = X_new.dot(self.w)
        return y_new
```

After this you can use, for example, grid search from sklearn:

```
from mysvm import mySVR

param_grid = {
    'C': [1e0, 1e1, 1e2, 1e3, 1e4, 1e5],
    'epsilon': [1e-2, 1e-1]
}
gs = GridSearchCV(mySVR(kernel='linear'), param_grid, cv=10, scoring='mean_squared_error'
    )
gs.fit(X, y)
print(gs.best_score_)
print(gs.best_params_)
```

4 Submission

When finished, submit your notebook (`.ipynb` file) and (if neccessary) the `.py` files which you use in your notebook. All the formulas and derivations must either be in markdown cells inside your notebook or in a separate PDF file. If your experiments lack an analysis or corresponding conclusions, your grade will be reduced!