Introduction to Algorithms for Data Mining and Machine Learning

Chapter 7: Support Vector Machine and Regression

Xin-She Yang

For details, please read the book:

Xin-She Yang, **Introduction to Algorithms for Data Mining and Machine Learning**, Academic Press/Elsevier, (2019).

SVM

The aim of classification is to try to separate different samples into different classes.

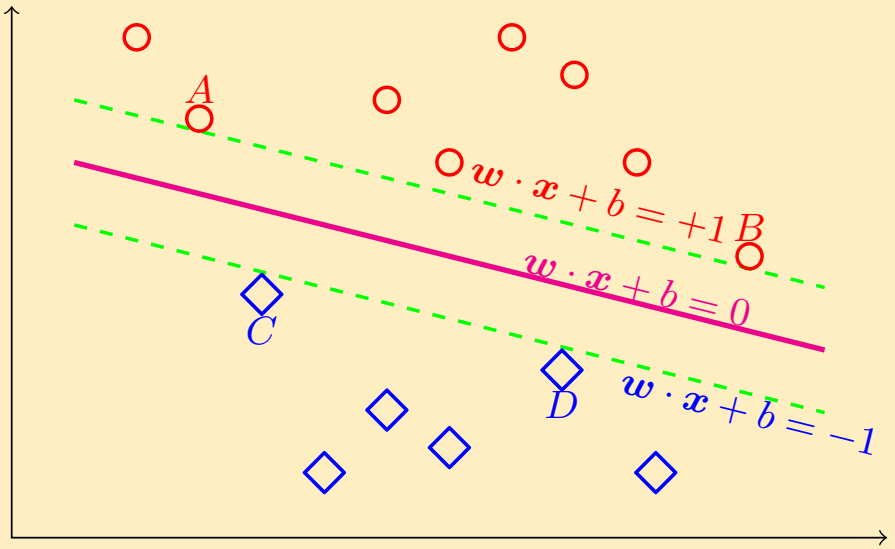
Binary Classification

For binary classification such as the diamonds and circles, we intend to construct a hyperplane (with a normal direction $\boldsymbol{w}$)

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0, \quad \text{or} \quad \boldsymbol{w}^T \boldsymbol{x} + b = 0,$$

so that these samples can be divided into two classes on two different sides.

This requires the existence of a hyperplane; otherwise, this approach will not work.



Four support vectors at four points (A,B,C,D).

In essence, if we can construct such a hyperplane, we should construct two hyperplanes (shown as dashed green lines) so that they are as far away as possible and no samples should be between these two planes. Mathematically, this is equivalent to two equations

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = +1, \quad \text{and} \quad \boldsymbol{w} \cdot \boldsymbol{x} + b = -1.$$

They can be written compactly as

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = \boldsymbol{w}^T \boldsymbol{x} + b = \pm 1.$$

Separation distance

The normal (perpendicular) distance d between these two (dashed green) hyperplanes is related to the norm $||\boldsymbol{w}||$ via

$$d = \frac{2}{||\boldsymbol{w}||}.$$

From the optimization point of view, the maximization of margins can be written as

$$\text{minimize} \quad \frac{1}{2}||\boldsymbol{w}||^2 = \frac{1}{2}||\boldsymbol{w}||_2^2 = \frac{1}{2}(\boldsymbol{w} \cdot \boldsymbol{w}).$$

If we can classify completely all samples $(\boldsymbol{x}_i, y_i)$ where $i = 1, 2, ..., n$, we have

$$\begin{aligned} \boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq +1, \quad & \text{if } (\boldsymbol{x}_i, y_i) \in \text{one class}, \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1, \quad & \text{if } (\boldsymbol{x}_i, y_i) \in \text{the other class}. \end{aligned}$$

As $y_i \in \{+1, -1\}$, the above two equations can be combined as

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, \quad (i = 1, 2, ..., n).$$

It is not always possible to construct such a separating hyperplane. However, we can use non-negative slack variables $\eta_i \geq 0$, $(i = 1, 2, ..., n)$ to relax the conditions, so that

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \eta_i, \quad (i = 1, 2, ..., n).$$

Linear SVM as an optimization problem

Now the optimization problem for the support vector machine (SVM) becomes

$$\text{minimize } \Psi = \frac{1}{2}||\boldsymbol{w}||^2 + \lambda \sum_{i=1}^{n} \eta_i,$$

subject to

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \eta_i, \quad \eta_i \geq 0, \quad (i = 1, 2, ..., n),$$

where $\lambda > 0$ is a penalty parameter. Here, the term $\sum_{i=1}^{n} \eta_i$ is essentially a measure of the upper bound of the number of misclassifications on the training data.

By using Lagrange multipliers $\alpha_i \geq 0$, we can rewrite the above constrained optimization into an unconstrained version, and we have

$$L = \frac{1}{2}||\boldsymbol{w}||^2 + \lambda \sum_{i=1}^{n} \eta_i - \sum_{i=1}^{n} \alpha_i [y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - (1 - \eta_i)].$$

From this, we can write the Karush-Kuhn-Tucker (KKT) conditions as

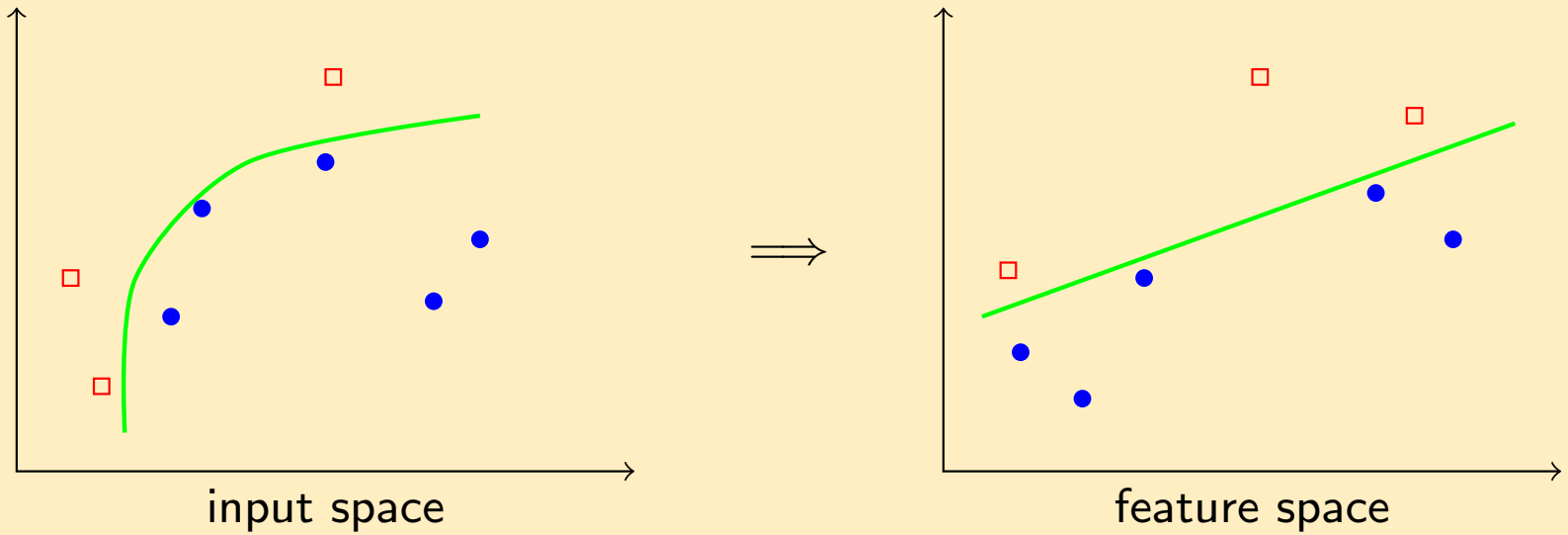
$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = 0, \quad \frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0, \quad y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - (1 - \eta_i) \geq 0,$$

$$\alpha_i [y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - (1 - \eta_i)] = 0, \quad \alpha_i \geq 0, \quad \eta_i \geq 0, \quad (i = 1, 2, ..., n).$$

From $\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = 0$, we get $\boldsymbol{w} = \sum_{i=1}^{n} y_i \alpha_i \boldsymbol{x}_i$. Only the corresponding training data $(\boldsymbol{x}_i, y_i)$ with $\alpha_i > 0$ can contribute to the solution, and thus such $\boldsymbol{x}_i$ form the support vectors (hence, the name support vector machine).

Nonlinear SVM

The SVM we discussed so far is linear because two classes can be separated by a hyperplane. In many applications, this is not possible. We have to use kernel tricks to transform the data in one space to another easier space.



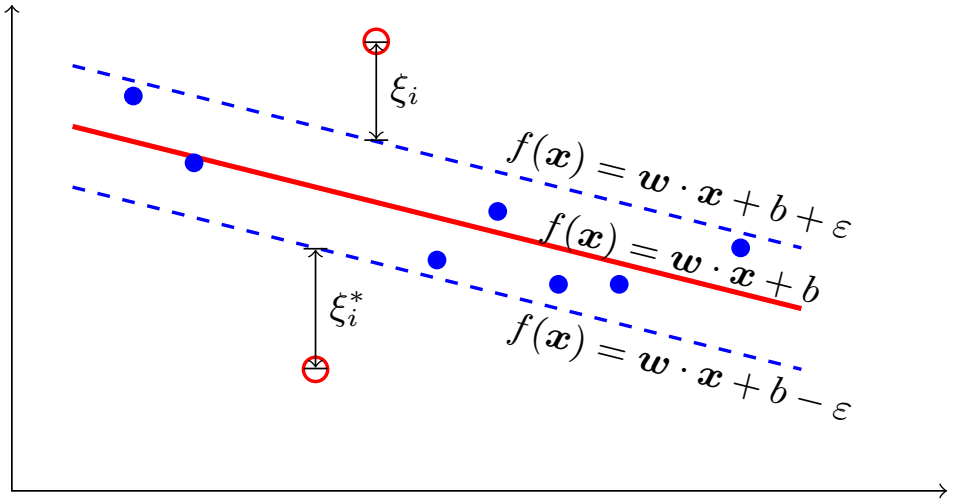
The most widely used kernel is the Gaussian radial basis function (RBF)

$$K(\boldsymbol{x}, \boldsymbol{x}_i) = \exp\Big[- ||\boldsymbol{x} - \boldsymbol{x}_i||^2/(2\sigma^2)\Big] = \exp\Big[- \gamma ||\boldsymbol{x} - \boldsymbol{x}_i||^2\Big] = \exp[-\gamma r^2],$$

for nonlinear classifiers. Here, $r = ||\boldsymbol{x} - \boldsymbol{x}_i||$ and γ is a hyperparameter.

Support Vector Regression

Support vector machines for classification work well for discrete labels. In case of a continuous dependent variable, it becomes a regression problem. Though SVM will not work, it can be modified to do regression, leading to support vector regression (SVR).



For a given tolerance ε, the ε-insensitive SVR is to find $y = f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = \boldsymbol{w} \cdot \boldsymbol{x} + b$ such that all the data points are within a strip bounded by two (blue) hyperplanes.

Mathematically, this regression problem can be written as

$$\text{Minimize } \frac{1}{2}||\boldsymbol{w}||^2, \quad \text{subject to} \quad |y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i - b| \le \varepsilon,$$

for all the n data point pairs $(\boldsymbol{x}_i, y_i)$ where $i = 1, 2, ..., n$. This is an optimization problem and in principle it can be solved using any proper optimization technique.

A relaxation extension is to allow some errors by using slack variables $\xi_i \geq 0$ (for the upper boundary) and $\xi_i^* \geq 0$ (for lower boundary) (see the figure). These slack variables must be non-negative ($\xi_i, \xi_i^* \geq 0$). The idea is to minimize $||\boldsymbol{w}||^2$ and also such errors.

SVR

The regression optimization problem can be written as

$$\text{Minimize } \frac{1}{2}||\boldsymbol{w}||^2 + \lambda \sum_i^n (\xi_i + \xi_i^*),$$

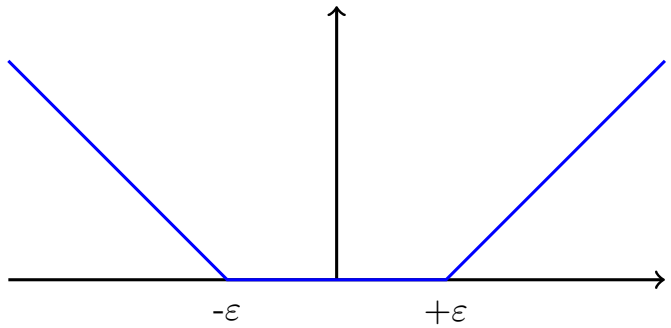
subject to

$$\begin{cases} y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i - b \leq \varepsilon + \xi_i, \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b - y_i \leq \varepsilon + \xi_i^*, \\ \xi_i \geq 0, \quad \xi_i^* \geq 0, \quad (i = 1, 2, ..., n). \end{cases}$$

Here the penalty parameter $\lambda > 0$ controls the tradeoff between the flatness of the regression model $f(\boldsymbol{x})$ and the deviation errors.

In the context of machine learning, we can use ε-insensitive loss function $|\xi|_\varepsilon$ for SVR

$$|\xi|_\varepsilon = \begin{cases} 0, & \text{if } |\xi| < \varepsilon, \\ |\xi| - \varepsilon, & \text{otherwise.} \end{cases}$$



We can analyze the SVR using KKT conditions and solve it using suitable optimization techniques. Both SVM and SVR have been implemented in many software packages.

References

References

- Xin-She Yang, **Introduction to Algorithms for Data Mining and Machine Learning**, Academic Press/Elsevier, (2019).
- Xin-She Yang, **Optimization Techniques and Applications with Examples**, John Wiley & Sons, (2018).

Notes on Software

Almost all major software packages in statistics, data mining and machine learning have implemented the algorithms we introduced here in this chapter.

- R has `svm`, while the Scikit-Learn for Python has `svm` with all algorithm variants.
- Matlab has `fitcsvm` and `fitrsvm` for classification and regression, respectively.
- In addition, SVM and SVR are also been implemented in C++, Java and other machine learning packages.

Any questions?

Thank you :)