Chapter 5. Support Vector Machines ¶

Power, versatile, linear & nonlinear classifications, regression, outlier detection.

Particularly well suited for classification of complex but small- or medium-sized datasets.

Linear SVM Classification

large margin classification: fitting the widest possible street (represented by the parallel dashed lines) between the classes.

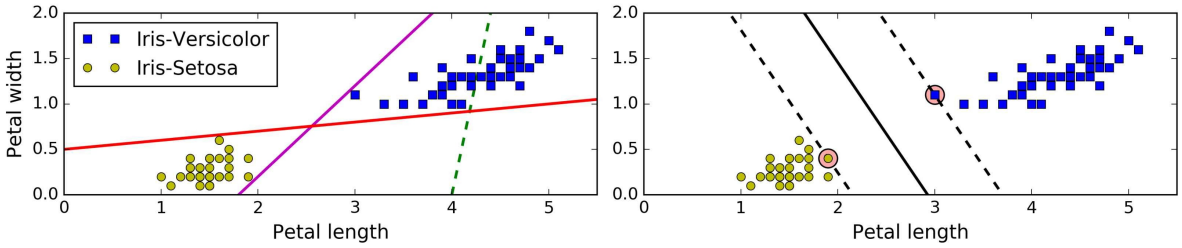

Figure 5-1. Large margin classification

Notice that adding more training instances “off the street” will not affect the decision boundary at all: it is fully determined (or "supported") by the instances located on the edge of the street. These instances are called the *support vectors*.

WARNING

> SVMs are sensitive to the feature scales.

Soft Margin Classification

Hard margin classification: if we strictly impose that all instances be off the street and on the right side.

Issues: 1) it only works if the data is linearly separable; 2) sensitive to outliers.

Soft margin classification: a good balance between keeping the street as large as possible and limiting the *margin violations*.

C hyperparameter: a smaller C value leads to a wider street but more margin violations, generalize better: fewer prediction errors.

TIP

> If your SVM model is overfitting, you can try regularizing it by reducing C .

NOTE

> Unlike Logistic Regression classifiers, SVM classifiers do not output probabilities for each class.

Nonlinear SVM Classification

One approach to handling nonlinear datasets is to add more features, such as polynomial features.

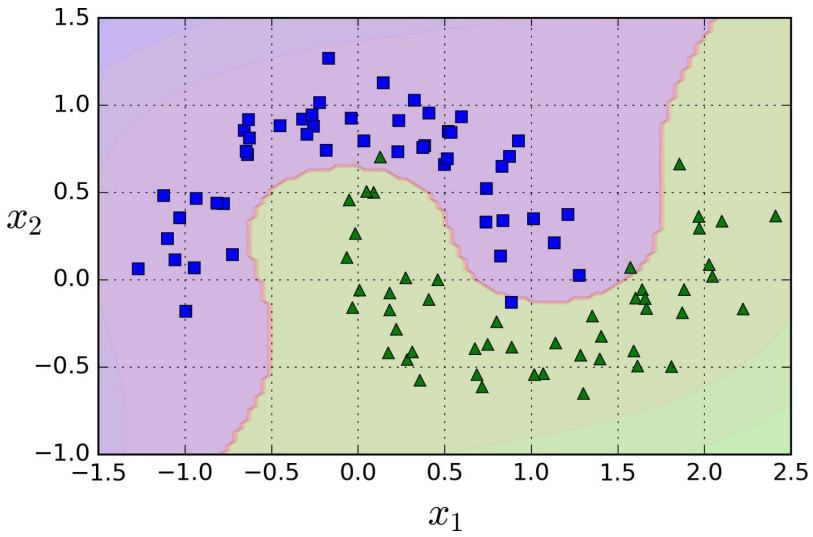

Figure 5-6. Linear SVM classifier using polynomial features

Polynomial Kernel

Kernel trick

Get the same result as if you added many polynomial features, even with very high-degree polynomials, without actually having to add them.

Adding Similarity Features

Another technique to tackle nonlinear problems is to add features computed using a *similarity function* that measures how much each instance resembles a particular *landmark*.

Gaussian Radial Basis Function (RBF)

$$\phi\gamma(\mathbf{x}, \iota) = exp(-\gamma\|\mathbf{x} - \iota\|^2)$$

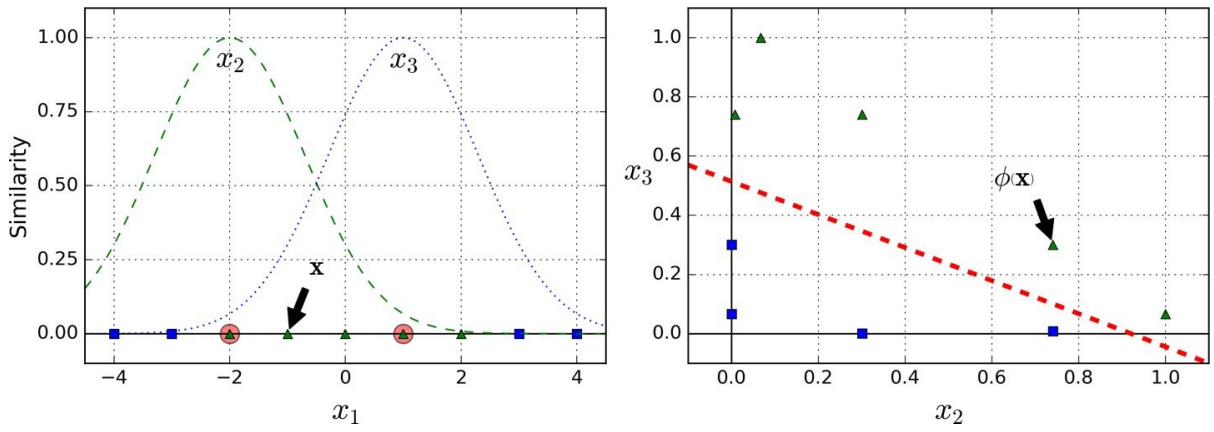

Figure 5-8. Similarity features using the Gaussian RBF

The simplest approach to select landmarks is to create a landmark at the location of each and every instance in the dataset. (# of features: $n \rightarrow m$)

Gaussian RBF Kernel

Apply kernel trick.

Hyperparameters gamma (γ) and C. Increasing gamma makes the bell-shape curve narrower, and as a result each instance's range of influence is smaller: the decision boundary ends up being more irregular, wiggling around individual instances.

TIP

> Which kernel to use? As a rule of thumb, try linear kernel first, especially if the training set is very large or if it has plenty of features. If the training set is not too large, you should try the Gaussian RBF kernel as well; it works well in most cases.

Computational Complexity

SVM regression

SVM Regression tries to fit as many instances as possible on the street while limiting margin violations. (i.e., instances *off* the street). The width of the street is controlled by a hyperparameter ϵ .

ϵ-insensitive model - adding more training instances within the margin does not affect the model's predictions;

NOTE

> SVMs can also be used for outlier detection.

Under the Hood

Decision Function and Predictions

Linear SVM classifier prediction

$$\hat{y} = \begin{cases} 0 \ \ if \mathbf{w}^T \cdot \mathbf{x} + b < 0 \\ 1 \ \ if \mathbf{w}^T \cdot \mathbf{x} + b \geq 0 \end{cases}$$

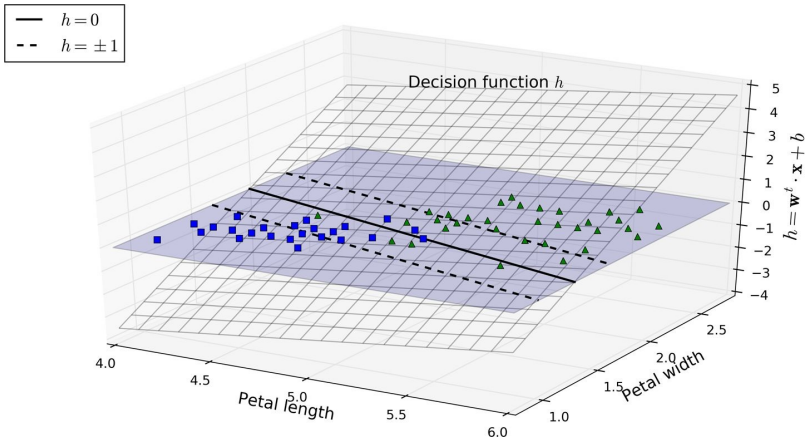

Figure 5-12. Decision function for the iris dataset

Training Objective

Minimize $||w||$ to get a large margin.

Hard margin linear SVM classifier objective

$$\underset{\mathbf{w},b}{minimize} \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w}$$
subject to $t^{(i)}(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b) \geq 1$ for $i = 1, 2, \cdots, m$

$t^{(i)} = -1$ for negative instances ($y^{(i)} = 0$) and $t^{(i)} = 1$ for positive instances ($y^{(i)} = 1$).

For soft margin, *slack variable* $\zeta^{(i)} \geq 0$ for each instance: $\zeta^{(i)}$ measures how much the i th instance is allowed to violate the margin.

Soft margin linear SVM classifier objective

$$\underset{\mathbf{w},b,\zeta}{minimize} \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} + C\sum_{i=1}^{m}\zeta^{(i)}$$
subject to $t^{(i)}(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b) \geq 1 - \zeta^{(i)}$ and $\zeta(i) \geq 0$ for $i = 1, 2, \cdots, m$

Quadratic Programming

The hard margin and soft margin problems are both convex quadratic optimization problems with linear constraints. Such problems are known as *Quadratic Programming* (QP) problems.

The Dual Problem

Given a constrained optimization problem, known as the *primal problem*, it is possible to express a different but closely related problem, called its *dual problem*.

The dual problem is faster to solve than the primal when the number of training instances is smaller than the number of features. More importantly, it makes the kernel trick possible, while the primal does not.

Kernelized SVM

Kernel trick for a 2nd-degree polynomial mapping

$$\phi(\mathbf{a})^T \cdot \phi(\mathbf{b}) = (\mathbf{a}^T \cdot \mathbf{b})^2$$

ϕ is the 2nd-degree polynomial mapping function.

2nd-degree *polynomial kernel*

$$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T \cdot \mathbf{b})^2$$

In Machine Learning, a *kernel* is a function capable of computing the dot product $\phi(\mathbf{a})^T \cdot \phi(\mathbf{b})$ based only on the original vectors $\mathbf{a}$ and $\mathbf{b}$, without having to compute the transformation ϕ.

Common Kernels:

Linear:

$$K(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \cdot \mathbf{b}$$

Polynomial:

$$K(\mathbf{a}, \mathbf{b}) = (\gamma \mathbf{a}^T \cdot \mathbf{b} + r)^d$$

Gaussian RBF:

$$K(\mathbf{a}, \mathbf{b}) = exp(-\gamma || \mathbf{a} - \mathbf{b} ||^2)$$

Sigmoid:

$$K(\mathbf{a}, \mathbf{b}) = tanh(\gamma \mathbf{a}^T \cdot \mathbf{b} + r)$$

Online SVMs

For linear SVM classifiers, one method is to use Gradient Descent to minimize the cost function

$$J(\mathbf{w}, b) = \frac{1}{2} \mathbf{w}^T \cdot \mathbf{w} + C \sum_{i=1}^{m} max(0, 1 - t^{(i)}(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b))$$

The second sum computes the total of all margin violations.

Hinge loss function

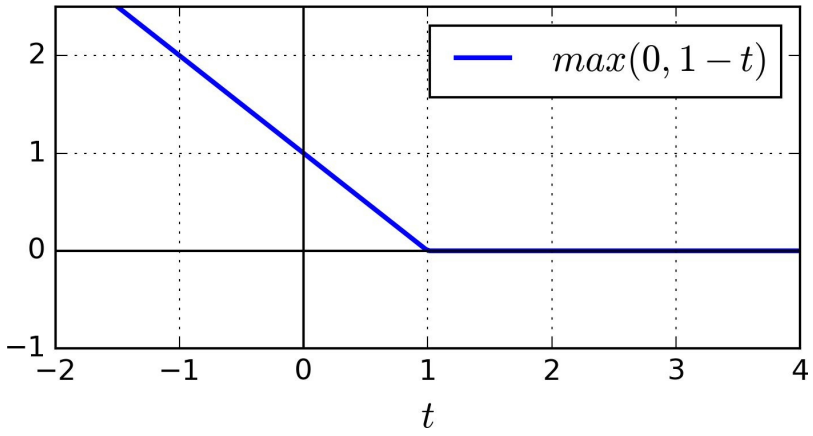

Figure. Hinge loss function

For large-scale nonlinear problems, consider using neural networks instead.