IT496: Introduction to Data Mining

Lecture 18

Support Vector Machines

(Slides are created from the book Hands-on ML by Aurelien Geron)

Arpit Rana

26th September 2023

Support Vector Machines

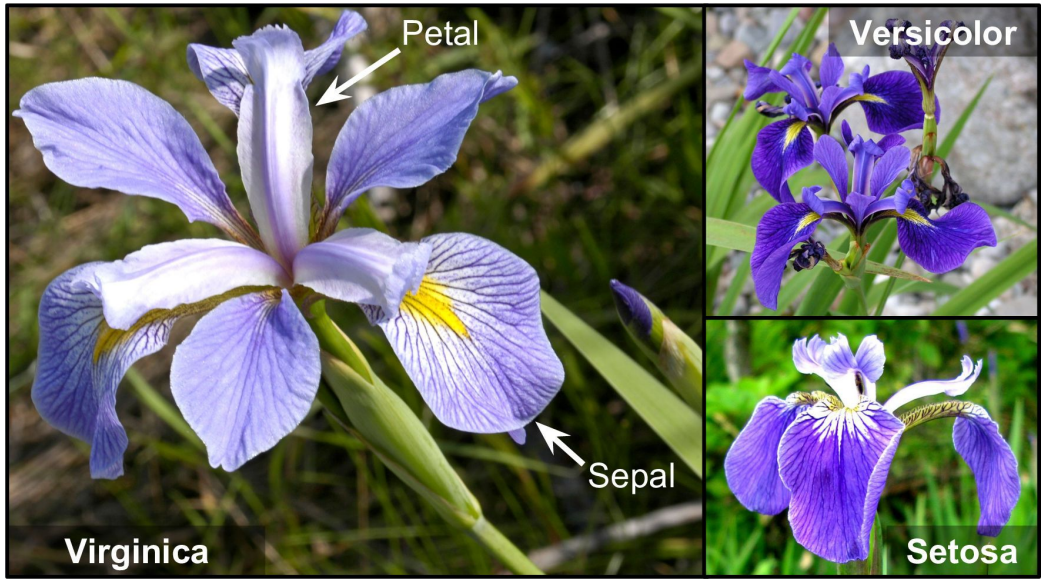
A Support Vector Machine (SVM) is a very powerful and versatile Machine Learning model, capable of performing

- linear or nonlinear classification,
- regression, and
- even outlier detection.

SVMs are particularly well suited for classification of complex but small- or medium-sized datasets.

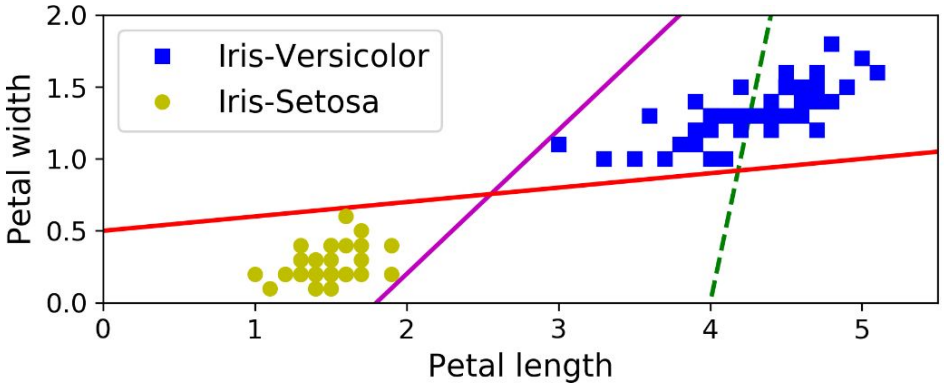
Iris Dataset

Iris flower image dataset that contains the *sepal* and *petal* length and *width* of 150 iris flowers of three different species: *Iris-Setosa*, *Iris-Versicolor*, and *Iris-Virginica*.



Classifying Iris Dataset

The two classes can clearly be separated easily with a straight line (they are *linearly separable*).

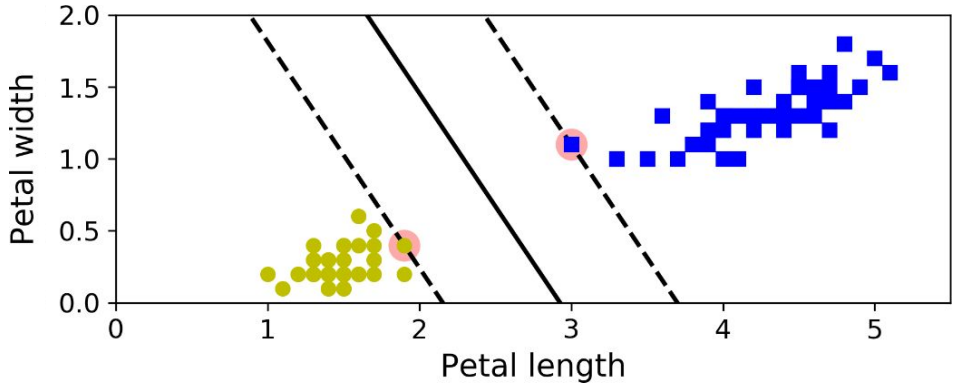


- The model whose decision boundary is represented by the green dashed line is so bad that it does not even separate the classes properly.
- The other two models work perfectly on this training set, but their decision boundaries come so close to the instances that these models will probably not perform as well on new instances.

Classifying Iris Dataset

The decision boundary of an SVM classifier not only separates the two classes but also stays as far away from the closest training instances as possible.

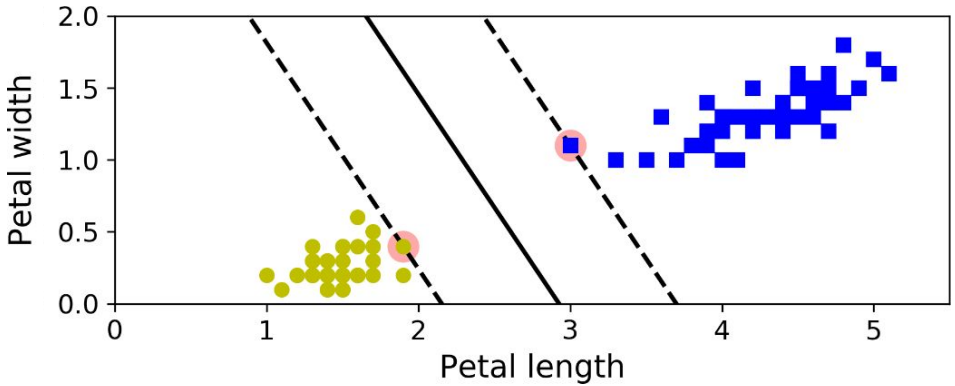
You can think of an SVM classifier as fitting the widest possible street (represented by the parallel dashed lines) between the classes.



This is called *large (maximum) margin classification.*

Classifying Iris Dataset

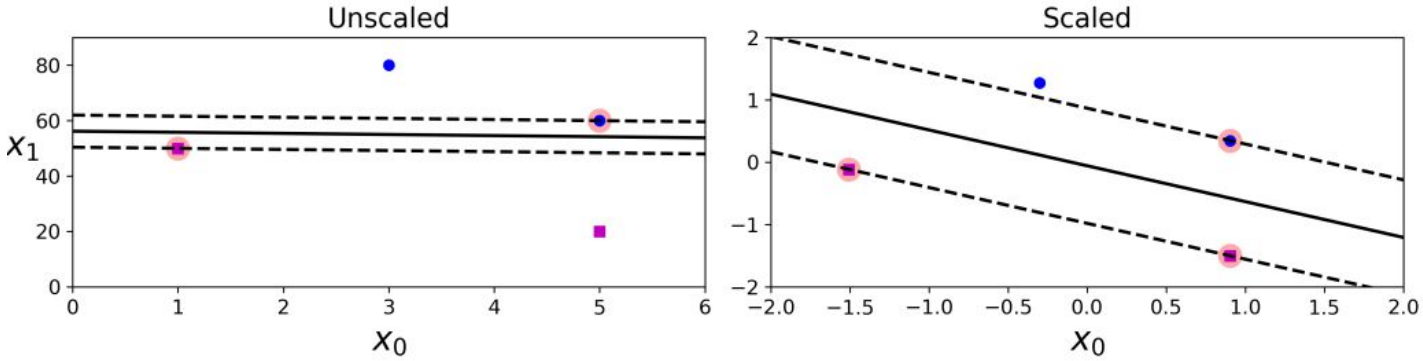
- Adding more training instances "*off the street*" will not affect the decision boundary at all.
- It is fully determined (or "supported") by the instances located on the edge of the street. These instances are called the ***support vectors***.



Classifying Iris Dataset

SVMs are sensitive to the feature scales (see in the figure below) -

- the vertical scale is much larger than the horizontal scale, so the widest possible street is close to horizontal.
- After feature scaling (e.g., using scikit-learn's `StandardScaler`), the decision boundary looks much better (on the right plot).

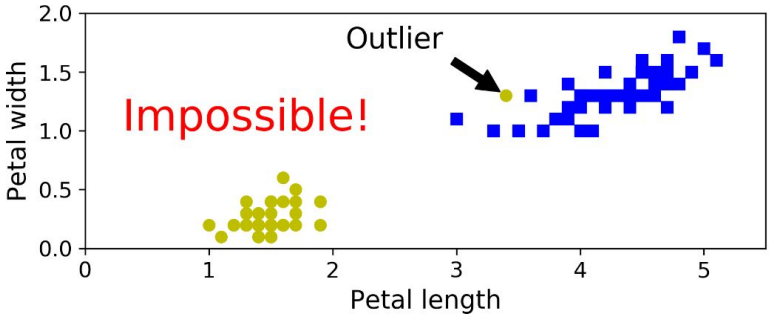


Hard Margin Classification

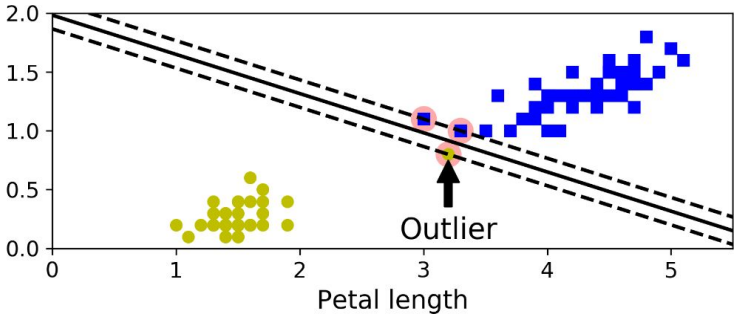
If we strictly impose that all instances be off the street and on the right side, this is called *hard margin classification*.

There are two main issues with hard margin classification.

- It only works if the data is linearly separable, and
- It is quite sensitive to outliers.



With just one additional outlier, it is impossible to find a hard margin



It will probably not generalize as well.

Soft Margin Classification

To avoid these issues it is preferable to use a more flexible model. The objective is to find a good balance between -

- keeping the street as large as possible and
- limiting the margin violations (i.e., instances that end up in the middle of the street or even on the wrong side).

This is called *soft margin classification*.

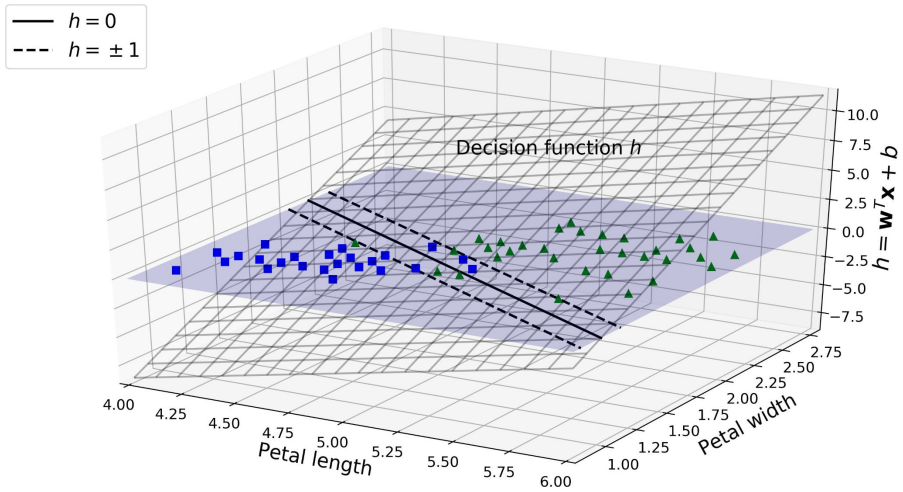
Decision Function and Prediction

The linear SVM classifier model predicts the class of a new instance $\boldsymbol{x}$ by simply computing the decision function -

$$w^T x + b = w_1 x_1 + w_2 x_2 + \ . \ . \ . \ + w_n x_n + b$$

If the result is positive, the predicted class $\hat{y}$ is the *positive class* (1), or else it is the *negative class* (0)

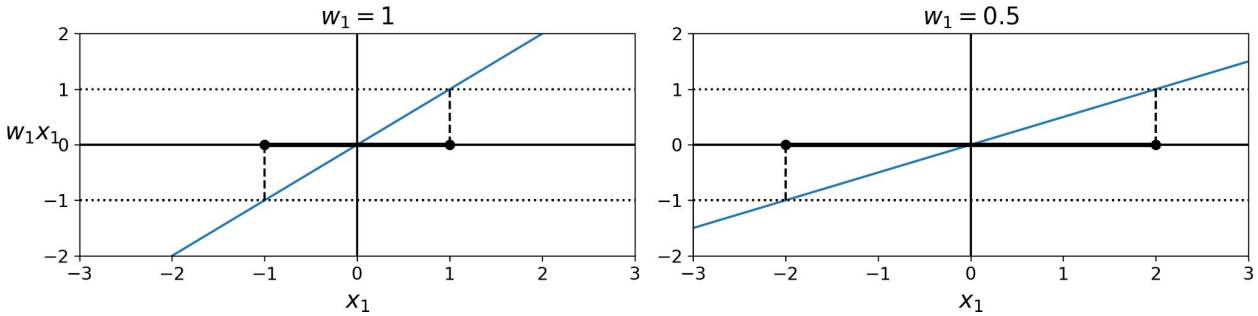
$$\hat{y} = \begin{cases} +1 & w^T x + b \geq 0, \\ -1 & w^T x + b < 0 \end{cases}$$



Training Objective

Training a linear SVM classifier means finding the value of w and b that make this margin as wide as possible while avoiding margin violations (hard margin) or limiting them (soft margin).

- Consider the slope of the decision function: it is equal to the norm of the weight vector, $\| w \|$.
- If we divide this slope by 2, the points where the decision function is equal to ±1 are going to be twice as far away from the decision boundary.



A smaller weight vector results in a larger margin

Hard Margin SVM

So, we want to minimize $\| w \|$ to get a large margin.

- If we also want to avoid any margin violation (hard margin), then we need the decision function to be greater than 1 for all positive training instances, and lower than −1 for negative training instances.
- We define $t^{(i)} = -1$ for negative instances (if $y^{(i)} = 0$) and $t^{(i)} = 1$ for positive instances (if $y^{(i)} = 1$), then we can express this constraint as $t^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1$ for all instances.

$$\underset{\mathbf{w}, b}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{subject to} \quad t^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \geq 1 \quad \text{for } i = 1, 2, \cdots, m$$

Hard Margin Linear SVM Classifier Objective

Soft Margin SVM

To get the soft margin objective, we need to introduce a *slack variable* $\zeta^{(i)} \geq 0$ for each instance: $\zeta^{(i)}$ measures how much the i^{th} instance is allowed to violate the margin.

We now have two conflicting objectives:

- making the slack variables as small as possible to reduce the margin violations, and
- making $1/2\ w^T w$ as small as possible to increase the margin.

$$\underset{\mathbf{w}, b, \zeta}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\zeta^{(i)}$$

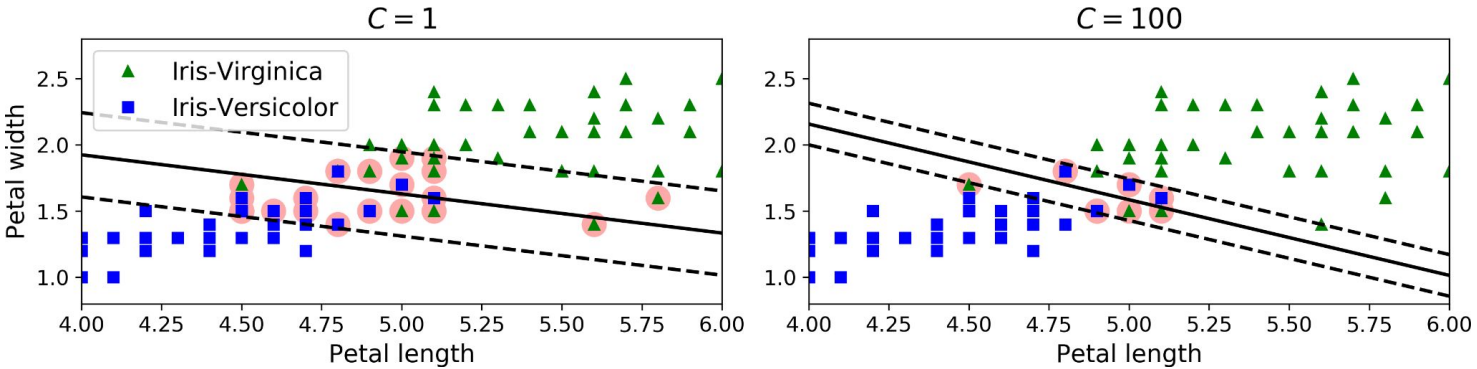
$$\text{subject to} \quad t^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \geq 1 - \zeta^{(i)} \quad \text{and} \quad \zeta^{(i)} \geq 0 \quad \text{for } i = 1, 2, \cdots, m$$

Soft Margin Linear SVM Classifier Objective

Soft Margin SVM

Here, C is a hyperparameter that defines the trade-off between the two objectives

- In Scikit-learn's SVM classes, you can control this balance using the C hyperparameter: a smaller C value leads to a wider street but more margin violations.
- If the SVM model is overfitting, you can try regularizing it by reducing C.



Large margin (left) versus fewer margin violations (right)

SVM in Python

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

```
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC

iris = datasets.load_iris()
X = iris["data"][:, (2, 3)]  # petal length, petal width
y = (iris["target"] == 2).astype(np.float64)  # Iris-Virginica

svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge")),
    ])

svm_clf.fit(X, y)

>>> svm_clf.predict([[5.5, 1.7]])
array([1.])
```

SVM in Python

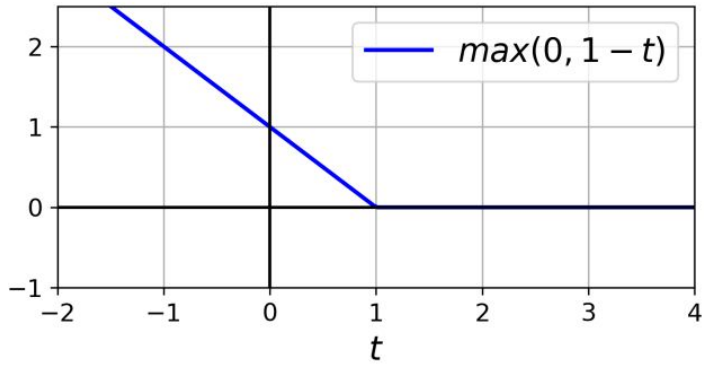
- Alternatively, we could use the `SVC` class, using `SVC(kernel="linear", C=1)`, but it is much slower, especially with large training sets.
- Another option is to use the `SGDClassifier` class, with `SGDClassifier(loss="hinge", alpha=1/(m*C))`.
    - It does not converge as fast as the `LinearSVC` class, but it can be useful to handle huge datasets that do not fit in memory (out-of-core training), or to handle online classification tasks.

Class	Time complexity	Out-of-core support	Scaling required	Kernel trick
`LinearSVC`	$O(m \times n)$	No	Yes	No
`SGDClassifier`	$O(m \times n)$	Yes	Yes	No
`SVC`	$O(m^2 \times n)$ to $O(m^3 \times n)$	No	Yes	Yes

Hinge Loss

The function `max(0, 1 - t)` is called the *hinge loss* function.

- It is equal to 0 when t ≥ 1. Its derivative (slope) is equal to −1 if t < 1 and 0 if t > 1.
- It is not differentiable at t = 1, but you can still use Gradient Descent using any subderivative at t = 1 (i.e., any value between −1 and 0).



Online SVM

Online SVM classifiers learn incrementally, typically as new instances arrive.

- For linear SVM classifiers, one method is to use Gradient Descent (e.g., using `SGDClassifier`) to minimize the cost function -

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

- The first sum in the cost function will push the model to have a small weight vector w, leading to a larger margin.
- The second sum computes the total of all margin violations.
- An instance's margin violation is equal to 0 if it is located off the street and on the correct side, or else it is proportional to the distance to the correct side of the street.

Next lecture

Non-linear SVM

28th September 2023