Machine Learning I (DATS 6202)

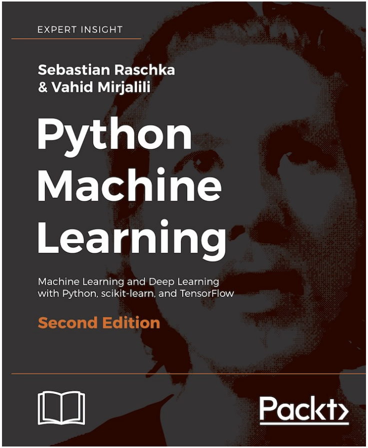
Support Vector Machines

Yuxiao Huang

Data Science, Columbian College of Arts & Sciences
George Washington University
yuxiaohuang@gwu.edu

October 30, 2018

Reference



Picture courtesy of the website of the book code repository and info resource

Reference

- This set of slices is an excerpt of the book by Raschka and Mirjalili, with some trivial changes by the creator of the slides
- Please find the reference to and website of the book below:
  - *Raschka S. and Mirjalili V. (2017). Python Machine Learning. 2nd Edition.*
  - `https://sebastianraschka.com/books.html`
- Please find the website of the book code repository and info resource below:
  - `https://github.com/rasbt/python-machine-learning-book-2nd-edition`

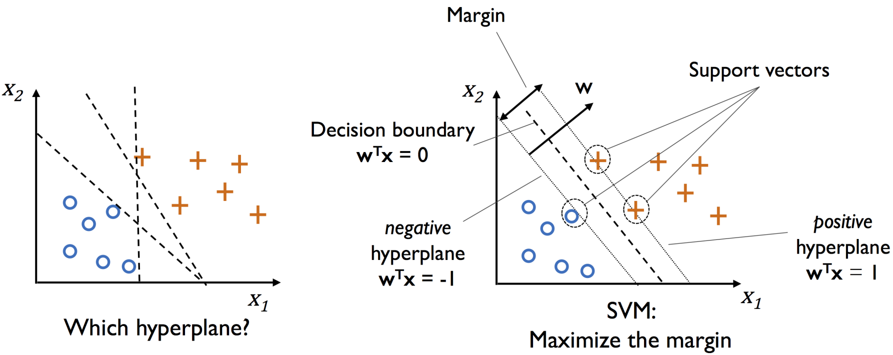
Overview

1. Intuition
2. The optimization approach
3. Estimating the parameters
4. Hard-margin and soft-margin classification
5. The kernel methods

Support vectors, margin, and support vector machines

- **Support vectors** are samples that are closest to the decision boundary
- **Margin** is the distance between the positive and negative hyperplane
- **Support vector machines** are classifiers that aim to maximize the margin
- Figure 1 (see next page) illustrates the concepts

Figure 1



x2
x1
Which hyperplane?
Margin
Support vectors
w
Decision boundary
wTx = 0
negative
hyperplane
wTx = -1
positive
hyperplane
wTx = 1
SVM:
Maximize the margin

The positive and negative hyperplanes

- The positive hyperplane:

$$w_0 + \mathbf{w}^T \mathbf{x}_{pos} = 1$$

- The negative hyperplane:

$$w_0 + \mathbf{w}^T \mathbf{x}_{neg} = -1$$

The margin

- When subtracting the previous two equations, we have

$$\mathbf{w}^T\left(\mathbf{x}_{pos} - \mathbf{x}_{neg}\right) = 2$$

- When normalizing the equation above by the length of the vector $\mathbf{w}$, $\|\mathbf{w}\|$, we have

$$\frac{\mathbf{w}^T\left(\mathbf{x}_{pos} - \mathbf{x}_{neg}\right)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \quad \text{where} \quad \|\mathbf{w}\| = \sqrt{\sum_{j=1}^{m} w_j^2}$$

- This equation represents the distance between the positive and negative hyperplane (a.k.a., the margin)

Estimating the parameters via optimization

- Again, the parameters can be estimated using optimization
- Here the objective function is the margin

$$\frac{2}{\|\mathbf{w}\|} \quad \text{where} \quad \|\mathbf{w}\| = \sqrt{\sum_{j=1}^{m} w_j^2}$$

- **Q:** Shall we maximize the margin or minimize it? Why?

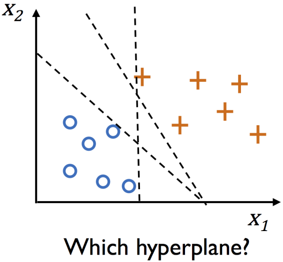
Estimating the parameters via optimization

- Again, the parameters can be estimated using optimization
- Here the objective function is the margin

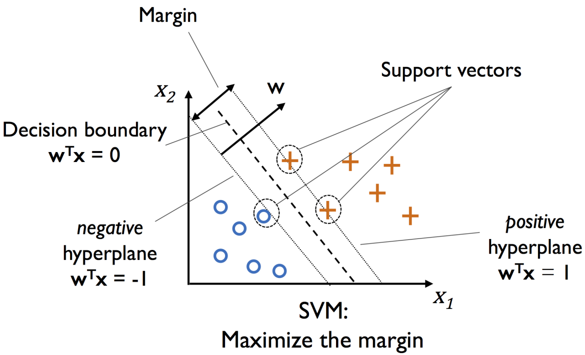
$$\frac{2}{\|\mathbf{w}\|} \quad \text{where} \quad \|\mathbf{w}\| = \sqrt{\sum_{j=1}^{m} w_j^2}$$

- **Q:** Shall we maximize the margin or minimize it? Why?
- **A:** Maximize it, since generally the larger the margin, the better the classification. This is shown in Figure 1 (see next page).

Figure 1



x_2
x_1
Which hyperplane?



Margin
Support vectors
x_2
w
Decision boundary
$w^T x = 0$
negative
hyperplane
$w^T x = -1$
positive
hyperplane
$w^T x = 1$
x_1
SVM:
Maximize the margin

The constraints

- **Q:** Is it true that the wider the margin the better?

The constraints

- **Q:** Is it true that the wider the margin the better?
- **A:** No, since a too wide margin may lead to misclassification
- Thus maximizing the margin should satisfy the constraint where the samples are classified correctly
- The constraint can be written as

$$\begin{cases} w_0 + \mathbf{w}^T \mathbf{x}_i \geq 1, & \text{if } y_i = 1 \\ w_0 + \mathbf{w}^T \mathbf{x}_i \leq -1, & \text{if } y_i = -1 \end{cases}$$

- The above two equations can be summarized as one

$$y_i \left(w_0 + \mathbf{w}^T \mathbf{x}_i \right) \geq 1$$

- **Q:** What does the equation mean?

The constraints

- **Q:** Is it true that the wider the margin the better?
- **A:** No, since a too wide margin may lead to misclassification
- Thus maximizing the margin should satisfy the constraint where the samples are classified correctly
- The constraint can be written as

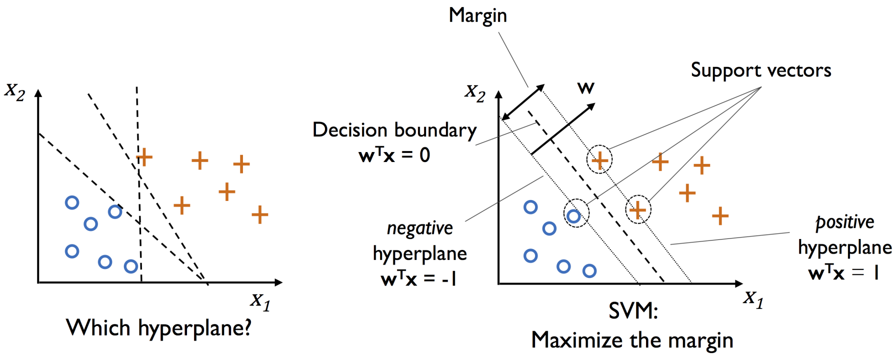
$$\begin{cases} w_0 + \mathbf{w}^T \mathbf{x}_i \geq 1, & \text{if } y_i = 1 \\ w_0 + \mathbf{w}^T \mathbf{x}_i \leq -1, & \text{if } y_i = -1 \end{cases}$$

- The above two equations can be summarized as one

$$y_i \left(w_0 + \mathbf{w}^T \mathbf{x}_i \right) \geq 1$$

- **Q:** What does the equation mean?
- **A:** The positive / negative samples should be behind the positive / negative hyperplane, as shown in Figure 1 (see next page)

Figure 1


x2
x1
Which hyperplane?
Margin
Support vectors
w
Decision boundary
wTx = 0
negative
hyperplane
wTx = -1
positive
hyperplane
wTx = 1
x2
x1
SVM:
Maximize the margin

The optimization problem

- The optimization entails:
  - finding $\mathbf{w}$ that maximizes the width (i.e., minimizes the reciprocal of the width):

$$\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|$$

  - finding $\mathbf{w}$ and w_0 that satisfy the constrain:

$$y_i\left(w_0 + \mathbf{w}^T\mathbf{x}_i\right) \geq 1$$

The lagrangian function

- The optimization can be converted into minimizing a lagrangian function

$$\mathcal{L}(\mathbf{w}, w_0, \alpha) = \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{margin: } \frac{2}{\|\mathbf{w}\|}} - \sum_i \alpha_i [\underbrace{y_i\big(w_0 + \mathbf{w}^T\mathbf{x}_i\big) - 1}_{\text{constrain: } y_i\left(w_0+\mathbf{w}^T\mathbf{x}_i\right)\geq 1}]$$

The lagrangian duality

- The optimization can be converted into minimizing a lagrangian function

$$\mathcal{L}(\mathbf{w}, w_0, \alpha) = \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{margin: } \frac{2}{\|\mathbf{w}\|}} - \sum_i \alpha_i [\underbrace{y_i\left(w_0 + \mathbf{w}^T\mathbf{x}_i\right) - 1}_{\text{constrain: } y_i\left(w_0 + \mathbf{w}^T\mathbf{x}_i\right) \geq 1}]$$

- Here:

$$(\mathbf{w}, w_0, \alpha) = \arg\max_{\alpha} \arg\min_{\mathbf{w}, w_0} \mathcal{L}(\mathbf{w}, w_0, \alpha) \quad \text{and}$$

$$(\mathbf{w}, w_0, \alpha) \text{ satisfy the KKT conditions}$$

The KKT conditions

- Condition 1:

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \alpha)}{\partial \mathbf{w}} = 0$$

- Condition 2:

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \alpha)}{\partial w_0} = 0$$

- Condition 3:

$$\alpha_i[y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1] = 0$$

- Condition 4:

$$y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \geq 0$$

- Condition 5:

$$\alpha_i \geq 0$$

The first order condition

- Based on conditions 1 and 2, we have

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- By substituting the equation into the lagrangian function, we have

$$\mathcal{L}(\mathbf{w}, w_0, \alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

Solving for α

- The value of α can be estimated by maximizing the lagrangian function:

$$\alpha = \arg\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

- This can be solved by quadratic programming, which is beyond the scope of this course

Solving for $\mathbf{w}$ and w_0

- Since the value of α is known, the value of $\mathbf{w}$ can be solved by:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

- Since the value of $\mathbf{w}$ is known, the value of w_0 can be solved by:

$$y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1 = 0$$

The decision rule

- Since the value of $\mathbf{w}$ and w_0 are known, the decision rule is known:

$$y_i = \begin{cases} 1, & \text{if } \mathbf{w}^T \mathbf{x}_i + w_0 \geq 0; \\ -1, & \text{otherwise} \end{cases}$$

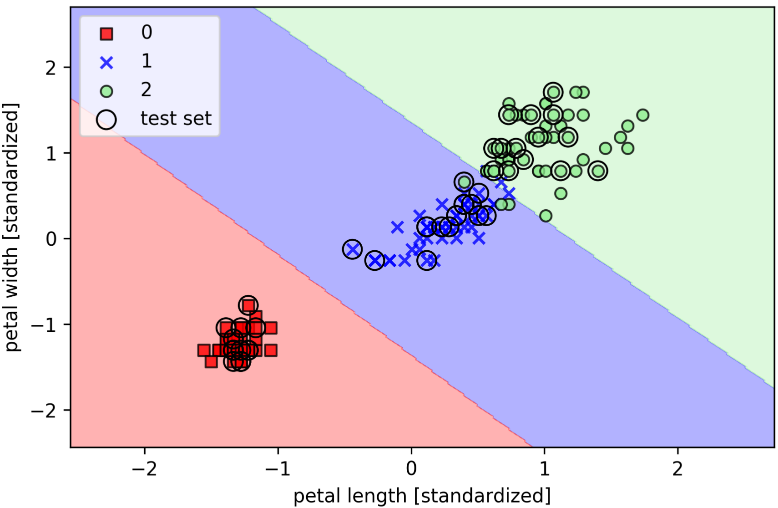
Linearly separable problems

- So far, we have been discussing the so-called **hard-margin classification**
- Here, the constraint for the optimization is

$$\begin{cases} w_0 + \mathbf{w}^T \mathbf{x}_i \geq 1, & \text{if } y_i = 1 \\ w_0 + \mathbf{w}^T \mathbf{x}_i < -1, & \text{if } y_i = -1 \end{cases}$$

  which says the positive / negative samples should be behind the positive / negative hyperplane
- Such constraint works well for linearly separable problems
- Figure 2 (see next page) shows that classes 0 (Setosa) and 1 (Versicolour) in Iris data are linearly separable

Figure 2


0
1
2
test set
2
1
0
−1
−2
petal width [standardized]
−2
−1
0
1
2
petal length [standardized]

Nonlinearly separable problems

- However, some problems are not linearly separable
- Figure 2 (see previous page) shows that classes 1 (Versicolour) and 2 (Virginica) in Iris data are not linearly separable
- The previous constraint does not work well for nonlinearly separable problems, since it does not allow misclassification

The slack variable

- To address nonlinearly separable problems, we can introduce a slack variable, ξ, to the constraint:

$$\begin{cases} w_0 + \mathbf{w}^T\mathbf{x}_i \geq 1 - \xi^{(i)}, & \text{if } y_i = 1 \\ w_0 + \mathbf{w}^T\mathbf{x}_i < -1 + \xi^{(i)}, & \text{if } y_i = -1 \end{cases}$$

- The new constraint says that, while most of the positive / negative samples should be behind the positive / negative hyperplane, some can be in front of it (thus allowing misclassification)
- The new constraint leads to the so-called **soft-margin classification**

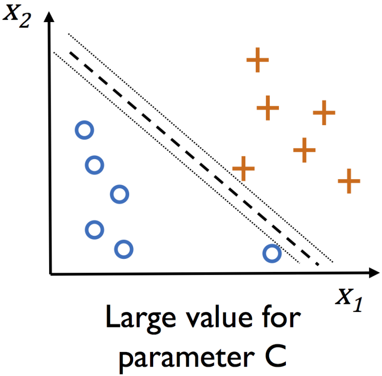
The new objective function

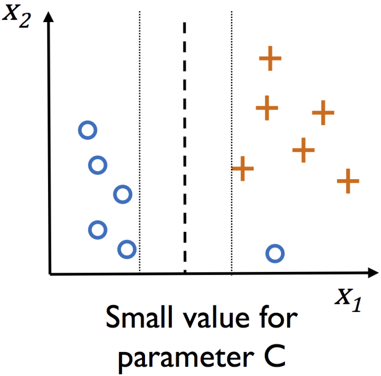
- With the new constraint, now the goal is to minimize the new objective function:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\Big(\sum_i \xi^{(i)}\Big)$$

  where C is used for penalizing large ξ
    - the larger the value of C, the larger the penalty
- As in logistic regression, C is used for regularization, which aims to tune the bias-variance trade-off
    - the larger the value of C, the smaller the bias while the larger the variance
    - the smaller the value of C, the larger the bias while the smaller the variance
- This is illustrated in figure 3 (see next page)

Figure 3


x_2
x_1
Large value for
parameter C

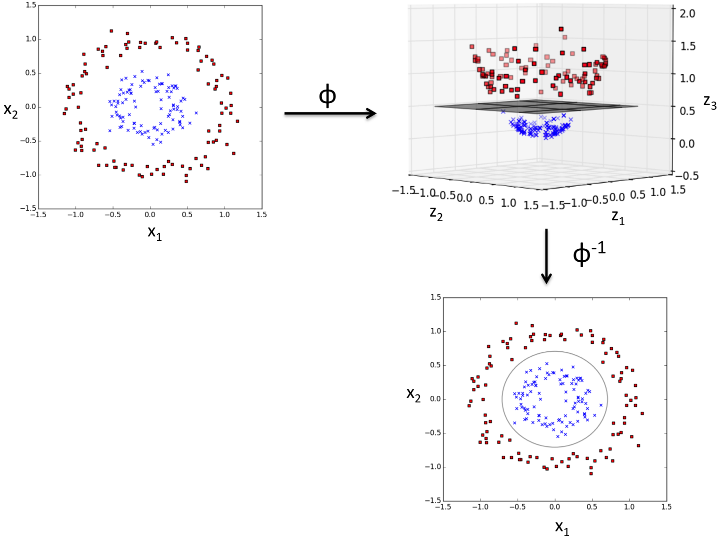

x_2
x_1
Small value for
parameter C

The kernel methods

- The idea of kernel methods is transforming a problem from a low dimension where linear separation does not exist, to a high dimension where linear separation does exist
- The transformation is done by a particularly designed kernel function, K, which creates nonlinear combinations of the features
- Figure 4 (see next page) shows an example of such transformation using the following mapping function

$$K(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$$

Figure 4



1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_2
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x_1
ϕ
2.0
1.5
1.0
0.5
0.0
−0.5
z_3
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
z_2
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
z_1
ϕ^{-1}
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_2
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x_1

Dot product

- Both the optimization and classification rely on the dot product of the features
- The lagrangian function:

$$\mathcal{L}(\mathbf{w}, w_0, \alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

- The decision rule:

$$y_j = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}_j + w_0 \geq 0; \\ -1, & \text{otherwise} \end{cases}$$

Dimension transformation

- The lagrangian function:

$$\mathcal{L}(\mathbf{w}, w_0, \alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{\mathbf{x}_i \cdot \mathbf{x}_j}_{\mathbf{x} \rightarrow \phi(\mathbf{x})}$$

- The decision rule:

$$y_j = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} \alpha_i y_i \underbrace{\mathbf{x}_i \cdot \mathbf{x}_j}_{\mathbf{x} \rightarrow \phi(\mathbf{x})} + w_0 \geq 0; \\ -1, & \text{otherwise} \end{cases}$$

Dimension transformation

- The lagrangian function:

$$\mathcal{L}(\mathbf{w}, w_0, \alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)}_{\mathbf{x} \to \phi(\mathbf{x})}$$

- The decision rule:

$$y_j = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} \alpha_i y_i \underbrace{\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)}_{\mathbf{x} \to \phi(\mathbf{x})} + w_0 \geq 0; \\ -1, & \text{otherwise} \end{cases}$$

The kernel

- The transformation from $\mathbf{x}_i \cdot \mathbf{x}_j$ to $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ is accomplished by the kernel, $K(\mathbf{x}_i, \mathbf{x}_j)$:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$

- Common kernels include:
  - Linear kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

  - Polynomial kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d$$

  - Radial basis function (RBF):

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\delta^2}} = e^{-r\|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

Kernel trick

- There are three steps for the transformation from $\mathbf{x}_i \cdot \mathbf{x}_j$ to $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$:

$$\mathbf{x}_i \cdot \mathbf{x}_j \rightarrow K(\mathbf{x}_i, \mathbf{x}_j) \rightarrow \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$

  1. choose a kernel K
  2. prove
  $$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$
  3. calculate the kernel

- The essence of kernel trick is that, we can use the steps above to accomplish the transformation in linear time without:
  - knowing what $\phi(\mathbf{x})$ is, which could be infinite
  - calculating $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$, which could be infeasible