Applied Data Science

Session 23, 24: Support Vector Machine

Dr. Soharab Hossain Shaikh

1

Mathematics Basics

Distance of a point (x_1,y_1) form a line **ax+by+c=0.**
$d = |ax_1+by_1+c|/\sqrt{(a^2+b^2)}$

Distance between two parallel lines:
L1 : ax + by + c_1=0.
L2 : ax + by + c_2=0.
$D = |c_1 - c_2|/\sqrt{(a^2+b^2)}$

Vector: w =[w1, w2]
Vector norm: $\sqrt{(w1^2+w2^2)}$

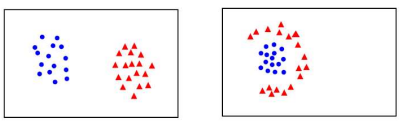
2

Binary Classifier

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots N$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(\mathbf{x})$ such that

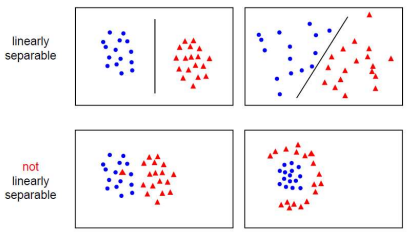
$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification.



3

Linear Separability

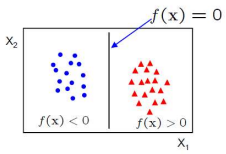


4

Linear Classifiers

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



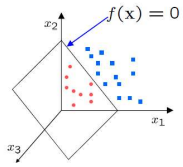
- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

5

Linear Classifiers

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

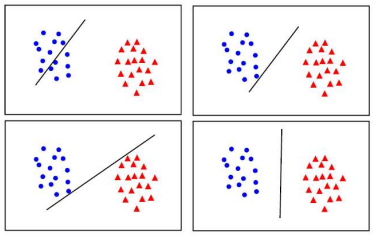


- in 3D the discriminant is a plane, and in nD it is a hyperplane

For a K-NN classifier it was necessary to `carry' the training data
For a linear classifier, the training data is used to learn **w** and then discarded
Only **w** is needed for classifying new data

6

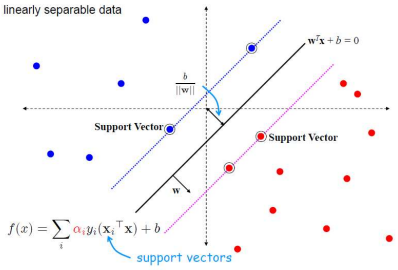
What is Best w?



- maximum margin solution: most stable under perturbations of the inputs

7

Support Vector Machine



8

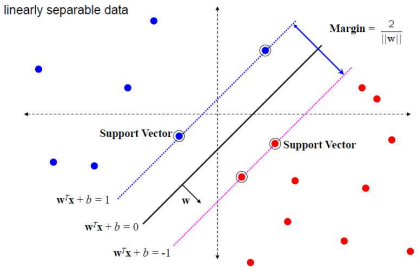
Derivation

- Since $\mathbf{w}^\top \mathbf{x} + b = 0$ and $c(\mathbf{w}^\top \mathbf{x} + b) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$
- Choose normalization such that $\mathbf{w}^\top \mathbf{x}_+ + b = +1$ and $\mathbf{w}^\top \mathbf{x}_- + b = -1$ for the positive and negative support vectors respectively
- Then the margin is given by

$$\frac{\mathbf{w}}{||\mathbf{w}||} \cdot (\mathbf{x}_+ - \mathbf{x}_-) = \frac{\mathbf{w}^\top (\mathbf{x}_+ - \mathbf{x}_-)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

9

Support Vector Machine



10

SVM Optimization

- Learning the SVM can be formulated as an optimization:

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{cases} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \text{ for } i = 1 \ldots N$$

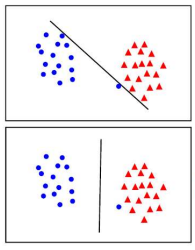
- Or equivalently

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ subject to } y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 \text{ for } i = 1 \ldots N$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

11

Linear Separability – What is best w?



• the points can be linearly separated but there is a very narrow margin

• but possibly the large margin solution is better, even though one constraint is violated

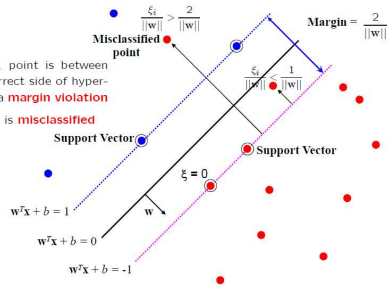
In general there is a trade off between the margin and the number of mistakes on the training data

12

Introduce Slack Variables

$\xi_i \geq 0$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**
- for $\xi > 1$ point is **misclassified**



13

"Soft" Margin Solution

The optimization problem becomes

$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i$$

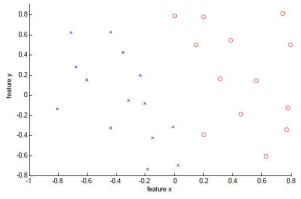
subject to

$$y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored → large margin
  - large C makes constraints hard to ignore → narrow margin
  - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C.

14

Soft Margin Classifier

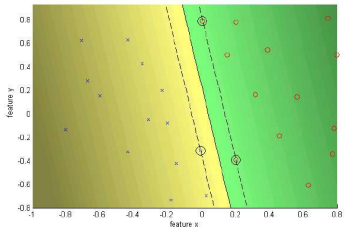


- data is linearly separable
- but only with a narrow margin

15

Soft Margin Classifier

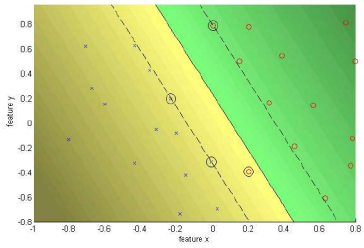
C = Infinity hard margin



16

Soft Margin Classifier

C = 10 soft margin



17

Optimization in SVM

Learning an SVM has been formulated as a constrained optimization problem over $\mathbf{w}$ and ξ

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i \text{ subject to } y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1\ldots N$$

The constraint $y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i$, can be written more concisely as

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i$$

which, together with $\xi_i \geq 0$, is equivalent to

$$\xi_i = \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

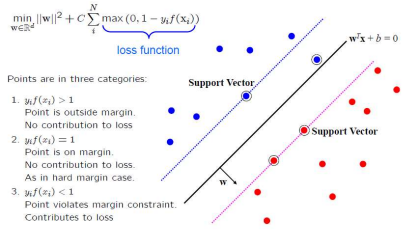
Hence the learning problem is equivalent to the unconstrained optimization problem over $\mathbf{w}$

$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

18

Loss Function

$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

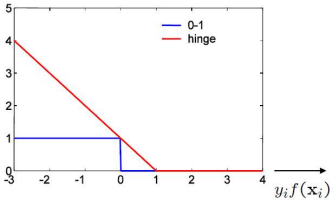


Points are in three categories:

1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss

19

Hinge Loss

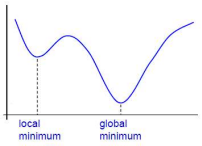


- SVM uses "hinge" loss $\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$
- an approximation to the 0-1 loss

20

Optimization in SVM

$$\min_{\mathbf{w}\in\mathbb{R}^d} C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i)) + ||\mathbf{w}||^2$$



- Does this cost function have a unique solution?
- Does the solution depend on the starting point of an iterative optimization algorithm (such as gradient descent)?

If the cost function is convex, then a locally optimal point is globally optimal (provided the optimization is over a convex set, which it is in our case)

21

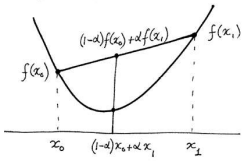
Convex Functions

D – a domain in $\mathbb{R}^n$.

A convex function $f : D \to \mathbb{R}$ is one that satisfies, for any $\mathbf{x}_0$ and $\mathbf{x}_1$ in D:

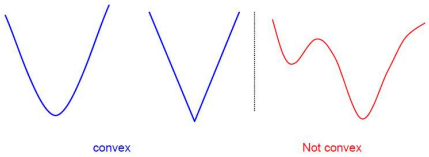
$$f((1-\alpha)\mathbf{x}_0 + \alpha\mathbf{x}_1) \leq (1-\alpha)f(\mathbf{x}_0) + \alpha f(\mathbf{x}_1) \ .$$

Line joining $(\mathbf{x}_0, f(\mathbf{x}_0))$ and $(\mathbf{x}_1, f(\mathbf{x}_1))$ lies above the function graph.



22

Convex Functions



A non-negative sum of convex functions is convex

23

Loss Functions in SVM

SVM

$$\min_{\mathbf{w}\in\mathbb{R}^d} C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i)) + ||\mathbf{w}||^2$$ convex

24

Gradient (Steepest) Descent Algorithm for SVM

To minimize a cost function $\mathcal{C}(\mathbf{w})$ use the iterative update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \mathcal{C}(\mathbf{w}_t)$$

where η is the learning rate.

First, rewrite the optimization problem as an average

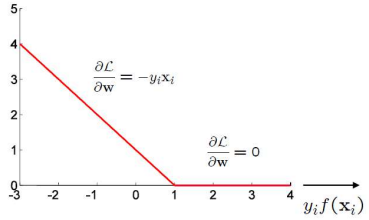
$$\begin{aligned} \min_{\mathbf{w}} \mathcal{C}(\mathbf{w}) &= \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i)) \\ &= \frac{1}{N}\sum_i^N \left(\frac{\lambda}{2}||\mathbf{w}||^2 + \max(0, 1 - y_i f(\mathbf{x}_i))\right) \end{aligned}$$

Because the hinge loss is not differentiable, a sub-gradient is computed

25

Sub-Gradient for Hinge Loss

$$\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) = \max(0, 1 - y_i f(\mathbf{x}_i)) \qquad f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$



26

Sub-Gradient Descent Algorithm for SVM

$$\mathcal{C}(\mathbf{w}) = \frac{1}{N}\sum_i^N \left(\frac{\lambda}{2}||\mathbf{w}||^2 + \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w})\right)$$

The iterative update is

$$\begin{aligned} \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta \nabla_{\mathbf{w}_t} \mathcal{C}(\mathbf{w}_t) \\ &\leftarrow \mathbf{w}_t - \eta \frac{1}{N}\sum_i^N (\lambda \mathbf{w}_t + \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{x}_i, y_i, \mathbf{w}_t)) \end{aligned}$$

where η is the learning rate.

Then each iteration t involves cycling through the training data with the updates:

$$\begin{aligned} \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta(\lambda \mathbf{w}_t - y_i \mathbf{x}_i) && \text{if } y_i f(\mathbf{x}_i) < 1 \\ &\leftarrow \mathbf{w}_t - \eta \lambda \mathbf{w}_t && \text{otherwise} \end{aligned}$$

27

So far we Have Discussed...

- We have seen that for an SVM learning a linear classifier

  $$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

  is formulated as solving an optimization problem over $\mathbf{w}$:

  $$\min_{\mathbf{w} \in \mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$$

- This quadratic optimization problem is known as the primal problem.
- Instead, the SVM can be formulated to learn a linear classifier

  $$f(\mathbf{x}) = \sum_i^N \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b$$

  by solving an optimization problem over α_i.
- This is know as the dual problem, and we will look at the advantages of this formulation.

28

Sketch of Derivation of the Dual Form

The Representer Theorem states that the solution $\mathbf{w}$ can always be written as a linear combination of the training data:

$$\mathbf{w} = \sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j$$

Now, substitute for $\mathbf{w}$ in $f(x) = \mathbf{w}^\top \mathbf{x} + b$

$$f(x) = \left(\sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j \right)^\top \mathbf{x} + b = \sum_{j=1}^{N} \alpha_j y_j \left(\mathbf{x}_j{}^\top \mathbf{x} \right) + b$$

and for $\mathbf{w}$ in the cost function $\min_{\mathbf{w}} ||\mathbf{w}||^2$ subject to $y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1, \forall i$

$$||\mathbf{w}||^2 = \left\{ \sum_j \alpha_j y_j \mathbf{x}_j \right\}^\top \left\{ \sum_k \alpha_k y_k \mathbf{x}_k \right\} = \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k)$$

Hence, an equivalent optimization problem is over α_j

$$\min_{\alpha_j} \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k) \quad \text{subject to } y_i \left(\sum_{j=1}^{N} \alpha_j y_j (\mathbf{x}_j{}^\top \mathbf{x}_i) + b \right) \geq 1, \forall i$$

and a few more steps are required to complete the derivation.

29

Primal and Dual Formulations

N is number of training points, and d is dimension of feature vector $\mathbf{x}$.

Primal problem: for $\mathbf{w} \in \mathbb{R}^d$

$$\min_{\mathbf{w} \in \mathbb{R}^d} ||\mathbf{w}||^2 + C \sum_{i}^{N} \max(0, 1 - y_i f(\mathbf{x}_i))$$

Dual problem: for $\boldsymbol{\alpha} \in \mathbb{R}^N$ (stated without proof):

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k) \quad \text{subject to } 0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and } \sum_i \alpha_i y_i = 0$$

- Need to learn d parameters for primal, and N for dual
- If $N << d$ then more efficient to solve for α than $\mathbf{w}$
- Dual form only involves $(\mathbf{x}_j{}^\top \mathbf{x}_k)$. We will return to why this is an advantage when we look at kernels.

30

Primal and Dual Formulations

Primal version of classifier:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

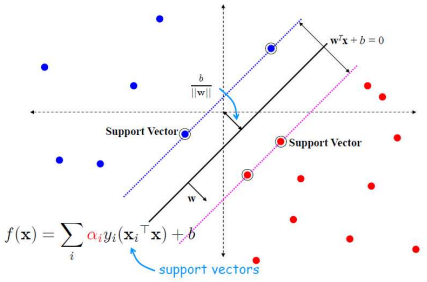
Dual version of classifier:

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

At first sight the dual form appears to have the disadvantage of a K-NN classifier – it requires the training data points $\mathbf{x}_i$. However, many of the α_i's are zero. The ones that are non-zero define the support vectors $\mathbf{x}_i$.

31

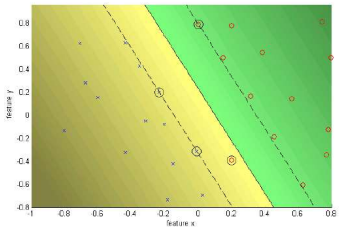
Support Vector Machine



32

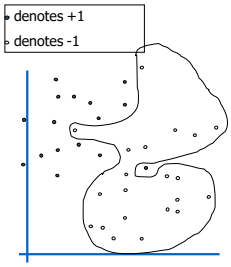
SVM – Soft Margin Classifier

C = 10 soft margin



33

Dataset with Noise



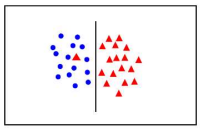
- **Hard Margin: Requires all data points be classified correctly**
  - **No training error**
- **What if the training set is noisy?**
  - **Solution 1: use very powerful kernels**

OVERFITTING!!!

- **Solution 2: use soft margin.**

34

Handling Data that is Not Linearly Separable

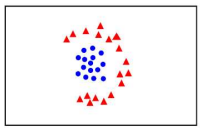


• introduce slack variables

$$\min_{w \in \mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||w||^2 + C \sum_i^N \xi_i$$

subject to

$$y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

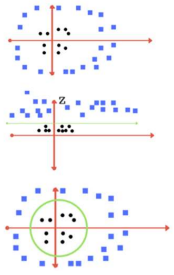


• linear classifier not appropriate

??

35

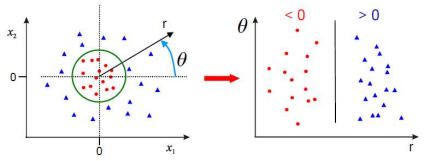
Not Linearly Separable



- Not separable by a linear decision boundary in the original feature planes.
- However, if **transformation is applied** to the data points (e.g. $z=x^2+y^2$), the data points become linearly separable in this transformed feature space.
- When we transform back this line to original plane, it maps to circular boundary.
- The learning of the hyper-plane in linear SVM is done by transforming the problem using linear algebra. This is where the **kernel** plays role.
- Polynomial and exponential kernels calculates separation line in higher dimension.

36

Feature Transformation



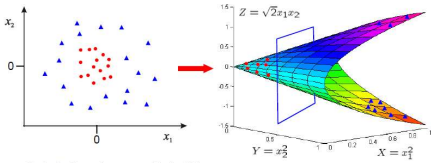
- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

37

Map Data to Higher Dimensional Space

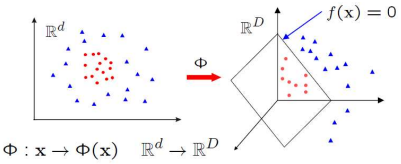
$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



- Data is linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

38

SVM Classifier in a Transformed Feature Space



$\Phi : \mathbf{x} \rightarrow \Phi(\mathbf{x}) \quad \mathbb{R}^d \rightarrow \mathbb{R}^D$

Learn classifier linear in $\mathbf{w}$ for $\mathbb{R}^D$:

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$$

$\Phi(\mathbf{x})$ is a feature map

39

Primal Classifier in Transformed Feature Space

Classifier, with $\mathbf{w} \in \mathbb{R}^D$:

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$$

Learning, for $\mathbf{w} \in \mathbb{R}^D$

$$\min_{\mathbf{w} \in \mathbb{R}^D} ||\mathbf{w}||^2 + C \sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$$

- Simply map $\mathbf{x}$ to $\Phi(\mathbf{x})$ where data is separable
- Solve for $\mathbf{w}$ in high dimensional space $\mathbb{R}^D$
- If $D >> d$ then there are many more parameters to learn for $\mathbf{w}$. Can this be avoided?

40

Dual Classifier in Transformed Feature Space

Classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b$$

$$\rightarrow f(\mathbf{x}) = \sum_i^N \alpha_i y_i \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}) + b$$

Learning:

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \mathbf{x}_j^\top \mathbf{x}_k$$

$$\rightarrow \max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_k)$$

subject to

$$0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and } \sum_i \alpha_i y_i = 0$$

41

Dual Classifier in Transformed Feature Space

- Note, that $\Phi(\mathbf{x})$ only occurs in pairs $\Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_i)$
- Once the scalar products are computed, only the N dimensional vector $\boldsymbol{\alpha}$ needs to be learnt; it is not necessary to learn in the D dimensional space, as it is for the primal
- Write $k(\mathbf{x}_j, \mathbf{x}_i) = \Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_i)$. This is known as a Kernel

Classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

Learning:

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k k(\mathbf{x}_j, \mathbf{x}_k)$$

subject to

$$0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and } \sum_i \alpha_i y_i = 0$$

42

Sub-Gradient Descent Algorithm for SVM

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\begin{aligned} \Phi(\mathbf{x})^\top \Phi(\mathbf{z}) &= \left(x_1^2, x_2^2, \sqrt{2} x_1 x_2\right) \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2} z_1 z_2 \end{pmatrix} \\ &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\ &= (x_1 z_1 + x_2 z_2)^2 \\ &= (\mathbf{x}^\top \mathbf{z})^2 \end{aligned}$$

Kernel Trick

- Classifier can be learnt and applied without explicitly computing $\Phi(\mathbf{x})$
- All that is required is the kernel $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^2$
- Complexity of learning depends on N (typically it is $O(N^3)$) not on D

43

Example Kernel

- Linear kernels $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- Polynomial kernels $k(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^\top \mathbf{x}'\right)^d$ for any $d > 0$
  - Contains all polynomials terms up to degree d
- Gaussian kernels $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$ for $\sigma > 0$
  - Infinite dimensional feature space

44

SVM Classifier with Gaussian Kernel

N = size of training data

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

α_i could be treated as the influence of the data point x_i on the final solution of $\boldsymbol{w}$.

weight (may be zero)

support vector

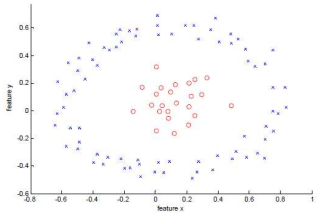
Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$

Radial Basis Function (RBF) SVM

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

45

SVM with RBF Kernel

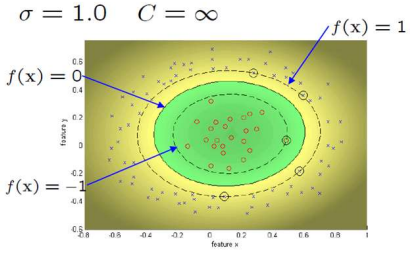


• data is not linearly separable in original feature space

46

SVM Classifier with Gaussian Kernel

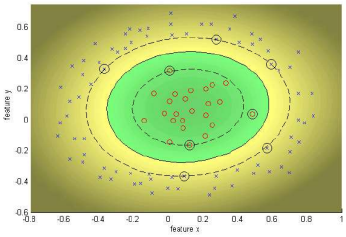
$\sigma = 1.0 \quad C = \infty$



47

SVM Classifier with Gaussian Kernel

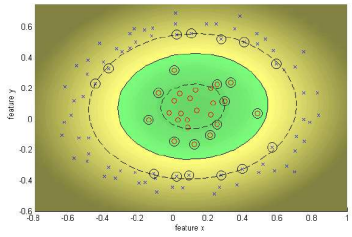
$\sigma = 1.0 \quad C = 100$



48

SVM Classifier with Gaussian Kernel

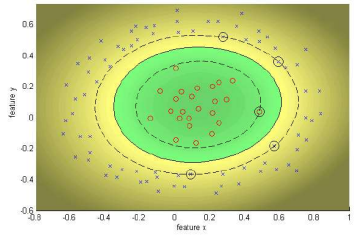
$\sigma = 1.0 \quad C = 10$



49

SVM Classifier with Gaussian Kernel

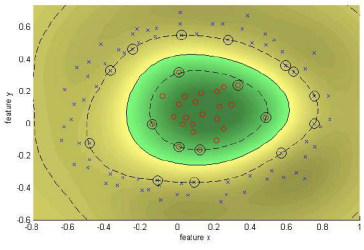
$\sigma = 1.0 \quad C = \infty$



50

SVM Classifier with Gaussian Kernel

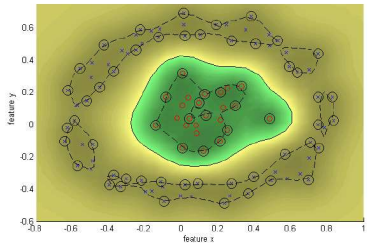
$\sigma = 0.25 \quad C = \infty$



51

SVM Classifier with Gaussian Kernel

$\sigma = 0.1 \quad C = \infty$



52

Kernel Trick

Feature Transformation in Higher Dimensional Space – one option could be that the data in raw representation have to be explicitly transformed into feature vector representations via a user-specified feature map.

In contrast, kernel methods require only a user-specified kernel, i.e., a similarity function over pairs of data points in raw representation.

Kernel methods owe their name to the use of kernel functions, which enable them to **operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space.**

This operation is often computationally cheaper than the explicit computation of the coordinates. This approach is called the "**kernel trick**".

Any linear model can be turned into a non-linear model by applying the kernel trick to the model: replacing its features (predictors) by a kernel function.

53

Kernel Trick

- The linear classifier relies on dot product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, the dot product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an *inner product* in some *expanded feature space*.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1\ x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1+\mathbf{x}_i^T\mathbf{x}_j)^2$,

  Need to show that $K(x_i,x_j)= \varphi(x_i)^T\varphi(x_j)$:

  $K(\mathbf{x}_i,\mathbf{x}_j)=(1+\mathbf{x}_i^T\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\,x_{i1}x_{j1}x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$

  $= [1\ \ x_{i1}^2\ \sqrt{2}\,x_{i1}x_{i2}\ \ x_{i2}^2\ \sqrt{2}x_{i1}\ \sqrt{2}x_{i2}]^T\,[1\ \ x_{j1}^2\ \sqrt{2}\,x_{j1}x_{j2}\ \ x_{j2}^2\ \sqrt{2}x_{j1}\ \sqrt{2}x_{j2}]$

  $= \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x}_j)$,

  where $\boldsymbol{\varphi}(\mathbf{x}) = [1\ \ x_1^2\ \sqrt{2}\,x_1x_2\ \ x_2^2\ \sqrt{2}x_1\ \sqrt{2}x_2]$

54

Nonlinear SVM

- SVM locates a separating hyper-plane in the feature space and classify points in that space.
- It does not need to represent the space explicitly, simply by defining a kernel function.
- The kernel function plays the role of the dot product in the feature space.

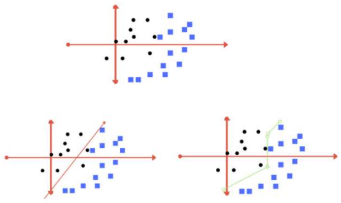
55

Kernel Trick Summary

- Classifiers can be learnt for high dimensional features spaces, without actually having to map the points into the high dimensional space.
- Data may be linearly separable in the high dimensional space, but not linearly separable in the original feature space.
- Kernels can be used for an SVM because of the scalar product in the dual form but can also be used elsewhere – they are not tied to the SVM formalism.

56

C – Regularization Parameter

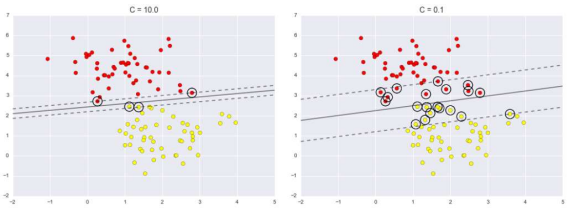


- C behaves as a regularization parameter in the SVM.
- It trades off correct classification of training examples against maximization of the decision function's margin.
- For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly.
- A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy.

Trades off misclassification of training examples against simplicity of the decision surface.

57

C – Regularization Parameter

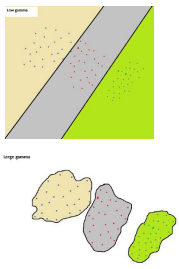


Low Bias & High Variance

High Bias & Low Variance

58

Gamma



One of the commonly used kernel functions is radial basis function (RBF). Gamma parameter of RBF controls the influence of distance of a single training point on the decision boundary.

Low values of gamma indicates a large similarity radius which results in more points being grouped together.

For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class).

Therefore, models with very large gamma values tend to overfit.

- The first image represents the case with a low gamma values. Similarity radius is large so all the points in the colored regions are considered to be in the same class. For instance, if we have a point the right bottom corner, it is classified as "green" class. On the other hand, the second image is the case with large gamma. For data points to be grouped in the same class, they must fall in the tight bounded area. Thus, a small noise may cause a data point to fall out of a class. Large gamma values are likely to end up in overfitting.
- As the gamma decreases, the regions separating different classes get more generalized. Very large gamma values result in too specific class regions (overfitting).

59

Weakness of SVM

- **It is sensitive to noise**
  - **A relatively small number of mislabeled examples can dramatically decrease the performance**
- **It only considers two classes**
  - **how to do multi-class classification with SVM?**
  - **Answer:**

1) with output arity *m*, learn m SVM's

- **SVM 1 learns "Output==1" vs "Output != 1"**
- **SVM 2 learns "Output==2" vs "Output != 2"**
- **:**
- **SVM m learns "Output==m" vs "Output != m"**

2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

60

Multi-class Classification

- **SVC** implements the "one-vs-one (ovo)" approach for multi- class classification. If ***n_class*** is the number of classes, then ***n_class * (n_class - 1) / 2*** classifiers are constructed and each one trains data from two classes. The final step of assigning a label to an unknown data-point is done by taking the majority voting.
- On the other hand, **LinearSVC** implements "**one-vs-rest (ovr)**" multi-class strategy, thus training ***n_class*** models (one model per class). If there are only two classes, only one model is trained (aka one-vs-all).
- **LinearSVC** also implements an alternative multi-class strategy, the so-called multi-class SVM formulated by Crammer and Singer, by using the option multi_class='crammer_singer'.
- In practice, one-vs-rest classification is usually preferred, since the results are mostly similar, but the runtime is significantly less.

61

Complexity

- Support Vector Machines are powerful tools, but their compute and storage requirements increase rapidly with the number of training vectors.
- The core of an SVM is a quadratic programming problem (QP), separating support vectors from the rest of the training data.
- The QP solver used by this libsvm-based implementation scales between $O(nfeatures \times nsamples^2)$ and $O(nfeatures \times nsamples^3)$ depending on how efficiently the libsvm cache is used in practice (dataset dependent).
- If the data is very sparse ***nfeatures*** should be replaced by the average number of non-zero features in a sample vector.

62

Properties of SVM

- Flexibility in choosing a similarity function.
- Sparseness of solution when dealing with large data sets - only support vectors are used to specify the separating hyper-plane.
- Ability to handle large feature spaces - complexity does not depend on the dimensionality of the feature space.
- Overfitting can be controlled by soft margin approach.
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution.

63

SVM Applications

- SVM has been used successfully in many real-world problems
  - -Text (and hypertext) categorization
  - -Image Classification
  - -Bioinformatics (Protein classification, Cancer classification)
  - -Hand-written character recognition

64

How to *build* a SVM Classifier Model in *Scikit-Learn*

65

Linear Classifier in Scikit-Learn

scikit-learn

`sklearn.svm`.LinearSVC

class `sklearn.svm.` **LinearSVC**(*penalty='l2', loss='squared_hinge', *, dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000*) [source]

- Build an estimator model like **LinearSVC()**
- Use **fit()** function to ***Train*** the model with **Training Dataset**
- Use **predict()** function to ***Test/Evaluate*** the model with **Test Dataset**
- Use **predict()** function to make ***prediction/inference*** on **New Unseen Data**

66

Example Code

```
>>> from sklearn.svm import LinearSVC
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import StandardScaler
>>> from sklearn.datasets import make_classification
>>> X, y = make_classification(n_features=4, random_state=0)
>>> clf = make_pipeline(StandardScaler(),
...                     LinearSVC(random_state=0, tol=1e-5))
>>> clf.fit(X, y)
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('linearsvc', LinearSVC(random_state=0, tol=1e-05))])
```

```
>>> print(clf.named_steps['linearsvc'].coef_)
[[0.141...   0.526... 0.679... 0.493...]]
```

```
>>> print(clf.named_steps['linearsvc'].intercept_)
[0.1693...]
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```

67

SVC in Scikit-Learn

scikit-learn

`sklearn.svm`.SVC

class `sklearn.svm.` **SVC**(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

- Build an estimator model like **SVC()**
- Use **fit()** function to ***Train*** the model with **Training Dataset**
- Use **predict()** function to ***Test/Evaluate*** the model with **Test Dataset**
- Use **predict()** function to make ***prediction/inference*** on **New Unseen Data**

68

Example Code

```
>>> import numpy as np
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import StandardScaler
>>> X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
>>> y = np.array([1, 1, 2, 2])
>>> from sklearn.svm import SVC
>>> clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
>>> clf.fit(X, y)
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svc', SVC(gamma='auto'))])
```

69

SVM Classifier – Not-Linearly Separable Data Points

`kernal` : linear, poly, rbf

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

70

71

Case-study

72

Case-study 1: Predicting Breast Cancer – Wisconsin Dataset

Objective :

Using historical data about patients who were diagnosed with cancer, enable the doctors to confidently differentiate malignant cases from benign given the independent attributes.

Prepare the data and create the model.

73

Case-study 1: Predicting Breast Cancer – Wisconsin Dataset

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)

Attribute Information:

1. Sample code number: id number
2. Clump Thickness: 1 - 10
3. Uniformity of Cell Size: 1 - 10
4. Uniformity of Cell Shape: 1 - 10
5. Marginal Adhesion: 1 - 10
6. Single Epithelial Cell Size: 1 - 10
7. Bare Nuclei: 1 - 10
8. Bland Chromatin: 1 - 10
9. Normal Nucleoli: 1 - 10
10. Mitoses: 1 - 10
11. Class: (2 for benign, 4 for malignant)

74

74

Case-study 2: Spam Filtering

Filtering messages – Spam/Ham & Hyper Parameter Tuning

We will talk about this case-study in **Text Processing** in **Module 4**.

75

Let's go to the Coding Demo…

76

To be continued in the next session…..

77