CS456: Machine Learning

Support Vector Machine + Kernel

Jakramate Bootkrajang

Department of Computer Science
Chiang Mai University

Objectives

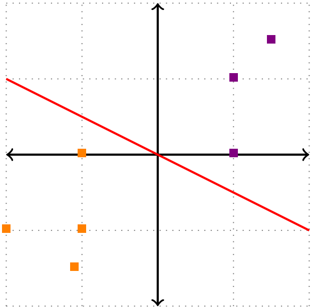
- To understand what decision boundary is
- To understand how SVM works and be able to apply SVM model effectively
- To understand what kernel is, and be able to choose appropriate kernel

Outlines

- Linear Decision Boundary
- SVM's philosophy
- Margin
- SVM + Kernel function
- Regularised SVM
- SVM in Sklearn

Linear decision boundary

A linear decision boundary is an imaginery line that separates data classes



How to draw such a line ?

- Consider a linear classifier

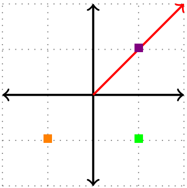
$$\text{sign}(\mathbf{w}^T\mathbf{x})$$

for binary classification where class label y is either 1 or -1

- This classifier predicts -1 if $\mathbf{w}^T\mathbf{x} < 0$
- and predicts 1 if $\mathbf{w}^T\mathbf{x} > 0$

Concrete example [1/3]

- For example, if $\mathbf{w} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$

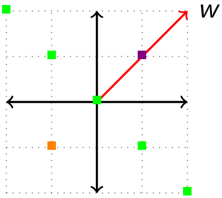


$$\mathbf{w} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

- We see that, $\mathbf{w}^T\mathbf{x}_{violet} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \end{bmatrix} > 0$ and $\mathbf{w}^T\mathbf{x}_{orange} < 0$

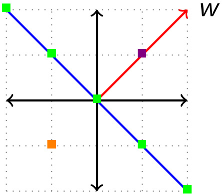
Concrete example [2/3]

- Now, observe that $\mathbf{w}^T\mathbf{x}_{green} = 0$.



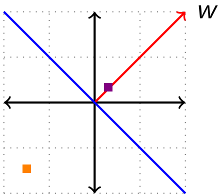
Concrete example [3/3]

- A set of points where $\mathbf{w}^T\mathbf{x} = 0$ defines the decision boundary.
- Geometrically, they are the points(vectors) which are perpendicular to $\mathbf{w}$. (dot product is zero)



Note that changing size of **w** does not change its decision boundary

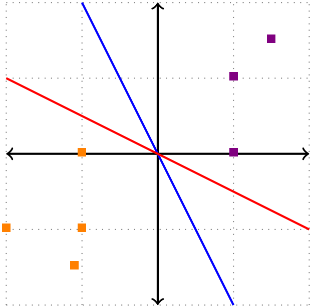
Decision boundary and prediction



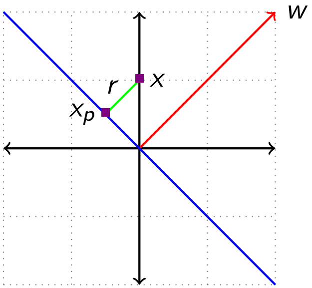
- Intuitively, we are more certain about the prediction of point that is far away from the boundary (orange point).
- while we are less sure about points lie closer to the decision boundary (violet point).

SVM's philosophy

- SVM makes use of such intuition and tries to find a decision boundary which is
  1. farthest away from data of both classes (maximum margin)
  2. predicts class labels correctly



Distance of x to the decision hyperplane



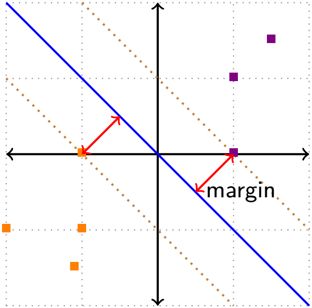
- Represent $\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{||\mathbf{w}||}$ (r $\times$ unit vector)
- Since $\mathbf{w}^T\mathbf{x}_p = 0$, we then have $\mathbf{w}^T\mathbf{x} = \mathbf{w}^T\mathbf{x}_p + \mathbf{w}^T r\frac{\mathbf{w}}{||\mathbf{w}||}$
- In other words, $r = \frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||}$, (note that r is invariant to scaling of $\mathbf{w}$.)

Margin

- The distance from the closest point in the class to decision boundary is called margin
- Usually, we assume margin of positive class and margin of negative class are equal
- This means decision boundary is placed in the middle of the distance between any two closest points having different class labels.

SVM's goal = maximum margin

According to a theorem from learning theory, from all possible linear decision functions the one that maximises the (geometric) margin of the training set will minimise the generalisation error



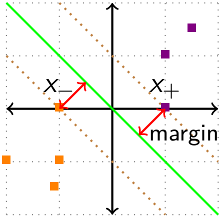
Types of margins

- Geometric margin: $r = \frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||}$
  - Normalised functional margin
- Functional margin: $\mathbf{w}^T\mathbf{x}$
  - Can be increased without bound by multiplying a constant to $\mathbf{w}$.

Maximum (geometric) margin (1/2)

- Since we can scale the functional margin, we can demand the functional margin for the nearest points to be $+1$ and -1 on the two side of the decision boundary.

- Denoting a nearest positive example by x_+ and a nearest negative example by x_-, we have
  - $w^T x_+ = +1$
  - $w^T x_- = -1$



Maximum (geometric) margin (2/2)

- We then compute the geometric margin from functional margin constraints

$$
\begin{aligned}
\text{margin} &= \frac{1}{2}\left(\frac{\mathbf{w}^T\mathbf{x}_+}{||\mathbf{w}||} - \frac{\mathbf{w}^T\mathbf{x}_-}{||\mathbf{w}||}\right) \\
&= \frac{1}{2||\mathbf{w}||}(\mathbf{w}^T\mathbf{x}_+ - \mathbf{w}^T\mathbf{x}_-) \\
&= \frac{1}{||\mathbf{w}||}
\end{aligned}
$$

SVM's objective function

- Given a *linearly separable* training set $S = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$.
- We need to find $\mathbf{w}$ which maximises $\frac{1}{||\mathbf{w}||}$
- Maximising $\frac{1}{||\mathbf{w}||}$ is equivalent to minimising $||\mathbf{w}||^2$
- The objective of SVM is then the following quadratic programming

$$\begin{aligned} \text{minimise:} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to:} \quad & \mathbf{w}^T\mathbf{x}_i \geq +1 \quad \text{for} \quad y_i = +1 \\ & \mathbf{w}^T\mathbf{x}_i \leq -1 \quad \text{for} \quad y_i = -1 \end{aligned}$$

SVM's philosophy: revisit

- SVM tries to find a decision boundary which is
  1. farthest away from data of both classes

$$\text{minimise: } \frac{1}{2}||\mathbf{w}||^2$$

  2. predicts class labels correctly

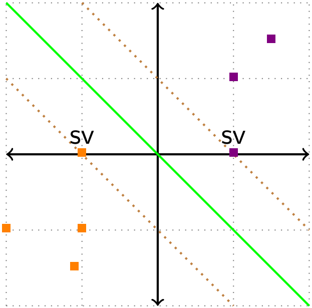
$$\begin{aligned}\text{subject to: } \mathbf{w}^T\mathbf{x}_i &\geq +1 \quad \text{for} \quad y_i = +1 \\ \mathbf{w}^T\mathbf{x}_i &\leq -1 \quad \text{for} \quad y_i = -1\end{aligned}$$

- We can write the above as:

$$\text{argmin}_{\mathbf{w}} \frac{1}{2}||\mathbf{W}||^2 + \sum_{i=1}^{N} \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i) \tag{1}$$

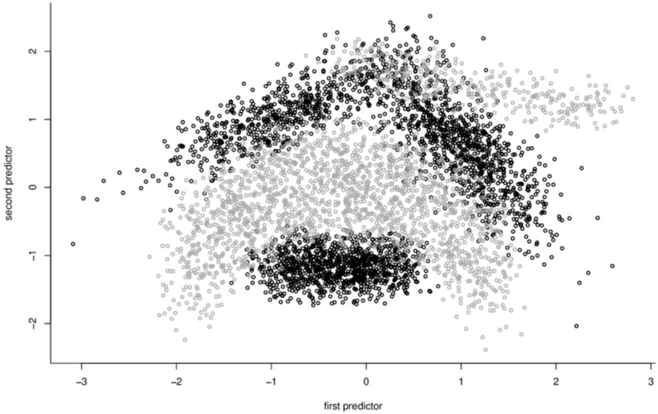
Support vectors

- The training points that are nearest to the decision boundary are called support vectors.
- Quiz: what is the output of our decision function for these points?



Non-linear data

What to do when data is not linearly separable?

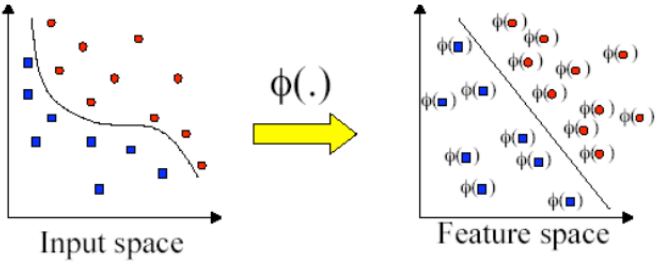


Non-linear classifiers

- First approach
  - Use non-linear model, e.g., neural network
  - (problems: many parameters, local minima)
- Second approach
  - Transform data into a richer feature space (including high dimensioal/non-linear features), then use a linear classifier

Learning in the feature space

Map data into a feature space where they are linearly separable



Feature transformation function

- It is computationally infeasible to explicitly compute the image of the mapping $\phi()$
- because some $\phi()$ even maps $\mathbf{x}$ into infinite dimensional space
- Instead of explicitly calculating $\phi(\mathbf{x})$ we can focus on the relative similarity between the mapped data
- That is, we calculate $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_i)$ for all i, j

Kernel

- A kernel is a function that gives the dot product between the vectors in feature space induced by the mapping ϕ.

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

- In a matrix form, a kernel matrix (a.k.a Gram matrix) is given by

$$K = \begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_j) & \dots & K(x_1, x_m) \\ \vdots & \ddots & \vdots & & \vdots \\ K(x_i, x_1) & \dots & K(x_i, x_j) & \dots & K(x_1, x_m) \\ \vdots & & \vdots & \ddots & \vdots \\ K(x_m, x_1) & \dots & K(x_m, x_j) & \dots & K(x_m, x_m) \end{bmatrix}$$

Kernel

- Each row of K is regarded as a trasformed data point, and K can be used in place of data matrix in training classification models

$$K = \begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_j) & \dots & K(x_1, x_m) \\ \vdots & \ddots & \vdots & & \vdots \\ K(x_i, x_1) & \dots & K(x_i, x_j) & \dots & K(x_1, x_m) \\ \vdots & & \vdots & \ddots & \vdots \\ K(x_m, x_1) & \dots & K(x_m, x_j) & \dots & K(x_m, x_m) \end{bmatrix}$$

Example 1: Polynomial kernel

- mapping $\mathbf{x}, \mathbf{y}$ points in 2D input to 3D feature space (note $\mathbf{y}$ is a data point not class label in this slide)

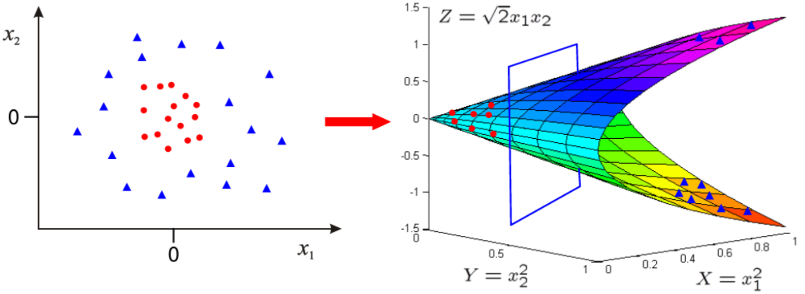
$$\mathbf{x}_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

- The corresponding mapping ϕ is

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \quad \phi(\mathbf{y}) = \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix}$$

- So we get a polynomial kernel $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T\phi(\mathbf{y}) = (\mathbf{x}^T\mathbf{y})^2$
  - of degree 2

Visualising



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

Figure: http://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf

- SVM with Polynomial kernel

  https://www.youtube.com/watch?v=3liCbRZPrZA

Kernels: Gaussian (rbf) kernel

- A Gaussian kernel is defined as

$$K(\mathbf{x}, \mathbf{y}) = e^{-||\mathbf{x}-\mathbf{y}||^2/\sigma}$$

- The mapping for Gaussian kernel $\phi()$ is of infinite dimensions
- σ is kernel parameter called 'kernel width', which must be chosen carefully
- Large σ assumes that neighbourhood of $\mathbf{x}$ spread further
- Small σ assumes small area of neighbourhood

Overlapping data ?

- We can not hope for every data being perfectly separable (both linearly and non-linearly)
- This includes naturally overlapping classes
- And also datasets which are quite noisy
  - Originally separable but due to some noise the observed data is not.
- If we try to apply SVM to this kind of data, there will be no solution
  - Since we require that all data must be correctly classified

Regularised SVM (1/3)

- We will relax constraint

$$\text{subject to: } \mathbf{w}^T\mathbf{x}_i \geq +1 \quad \text{for} \quad y_i = +1$$
$$\mathbf{w}^T\mathbf{x}_i \leq -1 \quad \text{for} \quad y_i = -1$$

- To

$$\text{subject to: } \mathbf{w}^T\mathbf{x}_i \geq +1 - \xi_i \quad \text{for} \quad y_i = +1$$
$$\mathbf{w}^T\mathbf{x}_i \leq -1 + \xi_i \quad \text{for} \quad y_i = -1$$

Regularised SVM (2/3)

- ξ_i s are called the slack variables
- Our new objective is then

$$\text{minimise:} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to:} \quad \mathbf{w}^T\mathbf{x}_i \geq +1 - \xi_i \;\; \text{for} \;\; y_i = +1$$

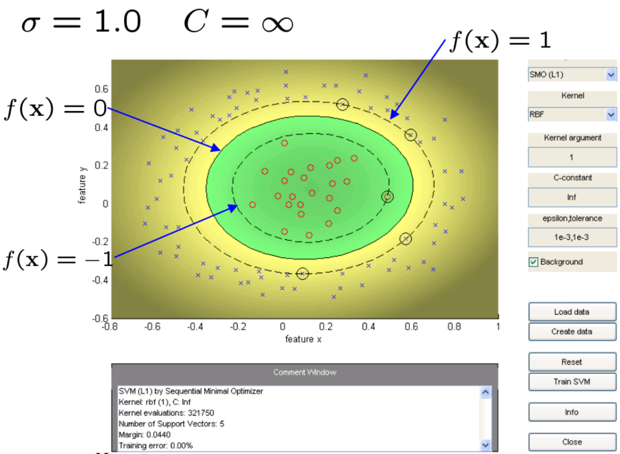
$$\mathbf{w}^T\mathbf{x}_i \leq -1 + \xi_i \;\; \text{for} \;\; y_i = -1$$

$$\xi_i \geq 0 \;\; \text{for all} \;\; i$$

Regularised SVM (3/3)

- Parameter C controls the trade-off between fitting the data well and allowing some slackness
- Predictive performance of SVM is known to depend on C paramater
  - Picking C usually done via cross-validation

Example of various C settings



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

Figure: http://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf

Example of various C settings

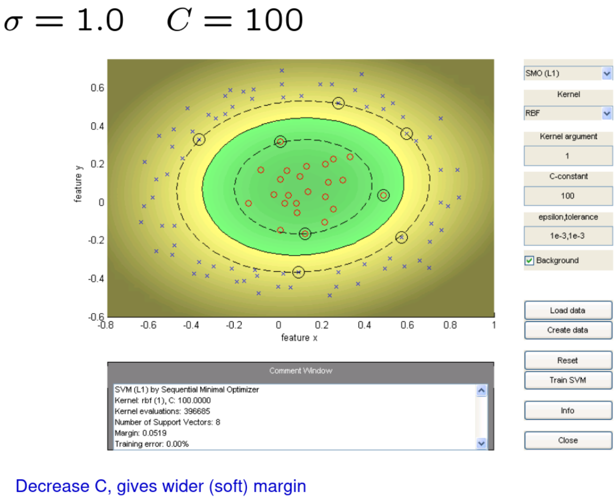
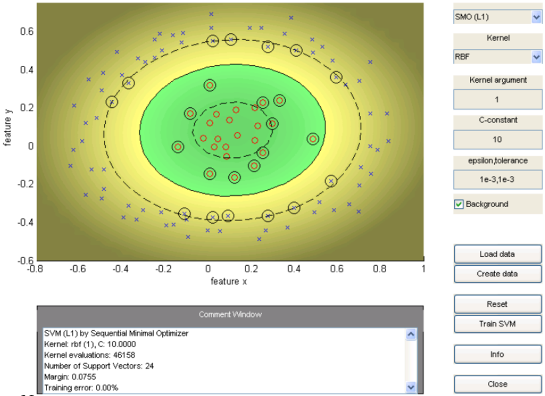


Figure: `http://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf`

Example of various C settings

$\sigma = 1.0 \quad C = 10$



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

Figure: http://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf

SVM in sklearn

Python's sklearn implements SVM in `svc` function

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*
cache_size=200, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape='ovr'*, *break_ties=False*,
random_state=None) [s

SVC parameters

We have seen all of the important parameters (gamma is σ in the slide)

Parameters:

C : ***float, optional (default=1.0)***

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel : ***string, optional (default='rbf')***

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : ***int, optional (default=3)***

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : ***{'scale', 'auto'} or float, optional (default='scale')***

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.

SVC example

An example of using SVC

```
import numpy as np
X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([1, 1, 2, 2])
from sklearn.svm import SVC
clf = SVC(gamma='auto')
clf.fit(X, y)

print(clf.predict([[-0.8, -1]]))
```

Objectives: revisited

- To understand what decision boundary is
- To understand how SVM works and be able to apply SVM model effectively
- To understand what kernel is, and be able to choose appropriate kernel

Lastly

Questions please ..