Support Vector Machines (SVMs)

Most of the slides have been taken from the following sources/slides:

R. Berwick, Village Idiot:
https://web.mit.edu/6.034/wwwbob/svm.pdf

Introduction

Support Vector Machine (SVM) is a popular supervised machine learning algorithm used for classification. It is particularly effective in solving complex problems with high-dimensional data.

Introduction

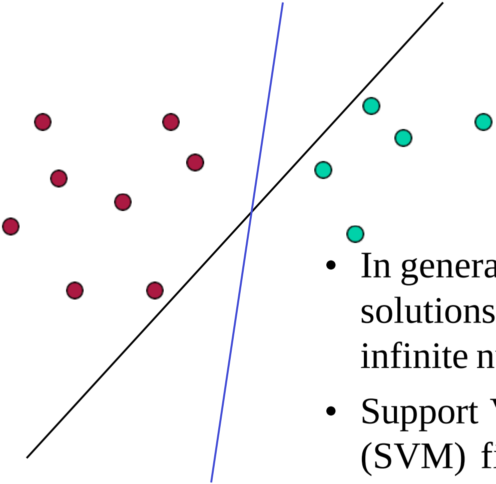
The basic idea behind SVM is to find an optimal hyperplane that separates the data points of different classes in the feature space. The hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest data points of each class. The data points closest to the hyperplane are called support vectors and play a crucial role in defining the decision boundary.

Support Vectors

- Support vectors are the data points that lie closest to the decision surface (or hyperplane)

- They are the data points most difficult to classify

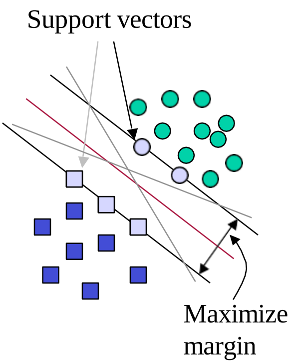
Which Separating Hyperplane?



- In general, lots of possible solutions for a,b,c (an infinite number!)
- Support Vector Machine (SVM) finds an <u>optimal</u> solution

Support Vector Machine (SVM)

- SVMs <u>maximize</u> the margin (Winston terminology: the 'street') around the separating hyperplane.
- The decision function is fully specified by a (usually very small) <u>subset</u> of training samples, the <u>support vectors.</u>
- This becomes a Quadratic programming problem that is easy to solve by standard methods



Separation by Hyperplanes

- Assume linear separability for now (we will relax this later)
- in 2 dimensions, can separate by a line
  - in higher dimensions, need hyperplanes

- For example, if a space is 3-dimensional then its hyperplanes are the 2-dimensional planes

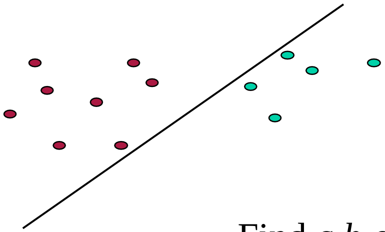
General input/output for SVMs just like for neural nets, but for one important addition…

Input: set of (input, output) training pair samples; call the input sample features $x_1, x_2 \ldots x_n$, and the output result y. Typically, there can be lots of input features x_i.

Output: set of weights **w** (or w_i), one for each feature, whose linear combination predicts the value of y. (So far, just like neural nets…)

Important difference: we use the optimization of maximizing the margin ('street width') to reduce the number of weights that are nonzero to just a few that correspond to the important features that 'matter' in deciding the separating line(hyperplane)…these nonzero weights correspond to the support vectors (because they 'support' the separating hyperplane)

2-D Case



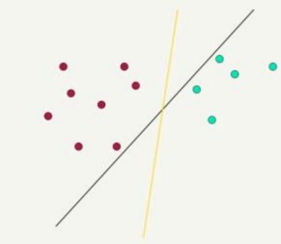
Find a,b,c, such that

$ax + by \geq c$ for red points

$ax + by \leq$ (or <) c for green points.

Which Hyperplane to pick?

- Lots of possible solutions for *a*,*b*,*c*.
- Some methods find a separating hyperplane, but not the optimal one (e.g., neural net)
- But: Which points should influence optimality?
  - All points?
    - Linear regression
  - Or only "difficult points" close to decision boundary
    - Support vector machines

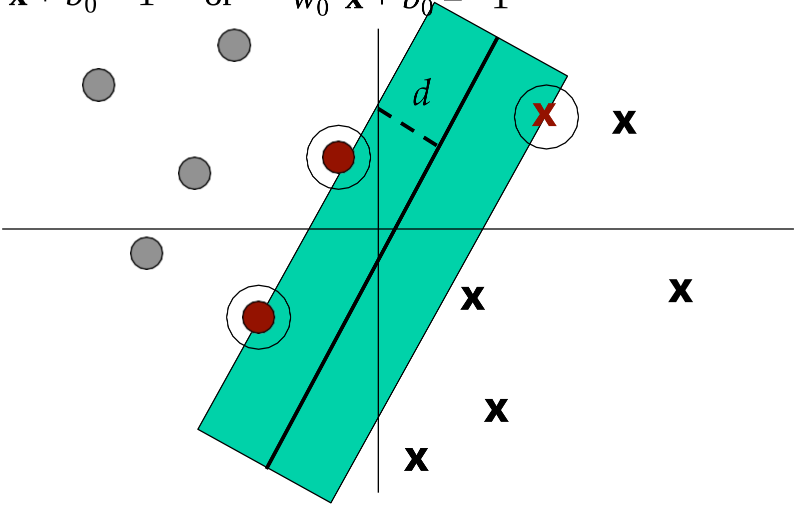


Support Vectors again for linearly separable case

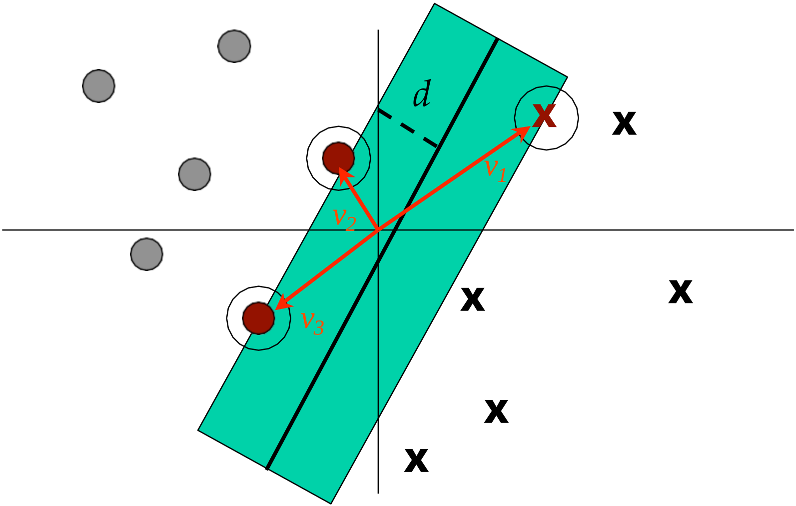
- Support vectors are the elements of the training set that would <u>change the position</u> of the dividing hyperplane if removed.
- Support vectors are the <u>critical</u> elements of the training set
- The problem of finding the optimal hyper plane is an optimization problem and can be solved by optimization techniques (we use Lagrange multipliers to get this problem into a form that can be solved analytically).

Support Vectors: Input vectors that just touch the boundary of the margin (street) – circled below, there are 3 of them

$w_0^T\mathbf{x} + b_0 = 1$ or $w_0^T\mathbf{x} + b_0 = -1$



Here, we have shown the actual support vectors, v_1, v_2, v_3, instead of just the 3 circled points at the tail ends of the support vectors. d denotes 1/2 of the street 'width'



Definitions

Define the hyperplanes H such that:

$w \bullet x_i + b \geq +1$ when $y_i = +1$

$w \bullet x_i + b \leq -1$ when $y_i = -1$

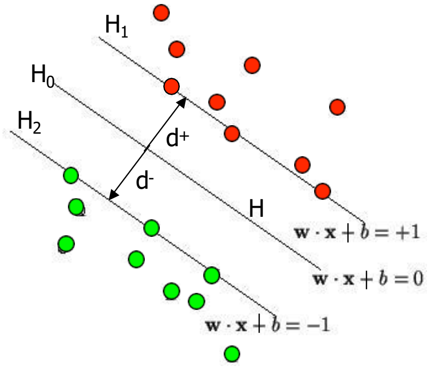
H_1 and H_2 are the planes:

H_1: $w \bullet x_i + b = +1$

H_2: $w \bullet x_i + b = -1$

The points on the planes H_1 and H_2 are the tips of the Support Vectors

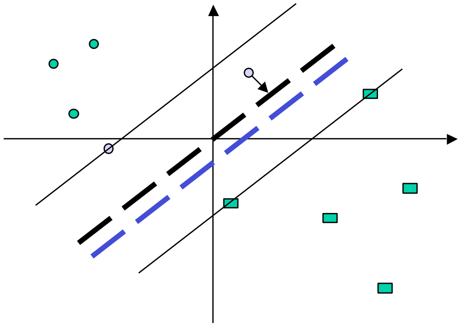
The plane H_0 is the median in between, where $w \bullet x_i + b = 0$



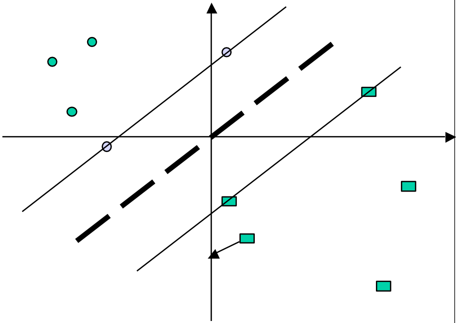
d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin (gutter) of a separating hyperplane is d+ + d–.



Moving a support vector moves the decision boundary



Moving the other vectors has no effect

The optimization algorithm to generate the weights proceeds in such a way that only the support vectors determine the weights and thus the boundary

Defining the separating Hyperplane

- Form of equation defining the decision surface separating the classes is a hyperplane of the form:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

  - **w is a weight vector**
  - **x is input vector**
  - **b is bias**
- Allows us to write

$$\mathbf{w}^T\mathbf{x} + b \geq 0 \text{ for } d_i = +1$$

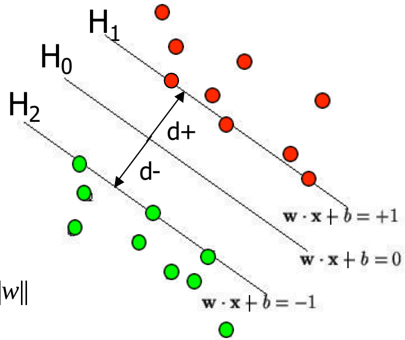
$$\mathbf{w}^T\mathbf{x} + b < 0 \text{ for } d_i = -1$$

Some final definitions

- Margin of Separation (*d*): the separation between the hyperplane and the closest data point for a given weight vector **w** and bias *b*.
- Optimal Hyperplane (maximal margin): the particular hyperplane for which the margin of separation *d* is maximized.

Maximizing the margin (aka street width)

We want a classifier (linear separator)
with as big a margin as possible.



Recall the distance from a point(x_0, y_0) to a line:
$Ax+By+c = 0$ is: $|Ax_0 + By_0 + c|/\text{sqrt}(A^2+B^2)$, so,
The distance between H_0 and H_1 is then:
$|w \bullet x+b|/\|w\|=1/\|w\|$, so

The total distance between H_1 and H_2 is thus: $2/\|w\|$

In order to maximize the margin, we thus need to minimize $\|w\|$. With the condition that there are no datapoints between H_1 and H_2:

$$\left.\begin{array}{l} \mathbf{x}_i \bullet \mathbf{w}+b \geq +1 \text{ when } y_i = +1 \\ \mathbf{x}_i \bullet \mathbf{w}+b \leq -1 \text{ when } y_i = -1 \end{array}\right\}$$

Can be combined into: $y_i(\mathbf{x}_i \bullet \mathbf{w} + b) \geq 1$

We now must solve a quadratic programming problem

- Problem is: minimize **||w||, s.t.** discrimination boundary is obeyed, i.e., min *f*(*x*) s.t. *g*(*x*)=0, which we can rewrite as: min *f*: ½ $||w||^2$ (Note this is a quadratic function)
  s.t. *g*: $\mathbf{y_i(x_i \cdot w + b) \geq 1}$

This is a **constrained optimization problem**

It can be solved by the Lagrangian multipler method

Because it is quadratic, the surface is a paraboloid, with just a single global minimum

How Langrangian solves constrained optimization

$$L(x,a) = f(x) + \sum_i a_i g_i(x)$$

The Lagrangian method: SVM

In our case, $f(x)$: $\frac{1}{2}\|\mathbf{w}\|^2$; $g(x)$: $y_i(\mathbf{w} \bullet x_i + b) - 1 = 0$ so Lagrangian is:

$$min\ L = \frac{1}{2}\|\mathbf{w}\|^2 - \Sigma a_i[y_i(\mathbf{w} \bullet x_i + \mathrm{b}) - 1] \text{ wrt } \mathbf{w},\ b$$

We expand the last to get the following L form:

$$min\ L = \frac{1}{2}\|\mathbf{w}\|^2 - \Sigma a_i y_i(\mathbf{w} \bullet x_i + \mathrm{b}) + \Sigma a_i \text{ wrt } \mathbf{w},\ b$$

Lagrangian Formulation

- So in the SVM problem the Lagrangian is

$$\min L_P = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} a_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} a_i$$

s.t. $\forall i, a_i \geq 0$ where l is the # of training points

- From the property that the derivatives at min = 0 we get:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} a_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L_P}{\partial b} = \sum_{i=1}^{l} a_i y_i = 0 \quad \text{so}$$

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

What is L_p?

- This indicates that this is the primal form of the optimization problem
- We will actually solve the optimization problem by now solving for the dual of this original problem
- What is this dual formulation?

The Lagrangian Dual Problem

The Lagrangian Dual Problem: instead of minimizing over **w**, *b*, subject to constraints involving *a*'s, we can maximize over *a* (the dual variable) subject to the relations obtained previously for **w** and *b*

Our solution must satisfy these two relations:

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

By substituting for **w** and *b* back in the original equation we can get rid of the dependence on **w** and *b*.

Primal problem:

$$\min L_P = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} a_i y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) + \sum_{i=1}^{l} a_i$$

s.t. $\forall i \ \mathrm{a}_i \geq 0$

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

Dual problem:

$$\max L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i=1}^{l} a_i a_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

s.t. $\sum_{i=1}^{l} a_i y_i = 0 \ \& \ a_i \geq 0$

(note that we have removed the dependence on **w** and *b*)

The Dual problem

- Kuhn-Tucker theorem: the solution we find here will be the same as the solution to the original problem
- Q: But why are we doing this???? (why not just solve the original problem????)
- Ans: Because this will let us solve the problem by computing the just the inner products of x_i, x_j (which will be very important later on when we want to solve non-linearly separable classification problems)

The Dual problem

Dual problem:

$$\max L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2}\sum_{i=1}^{l} a_i a_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

$$\text{s.t.} \ \sum_{i=1}^{l} a_i y_i = 0 \ \& \ a_i \geq 0$$

Notice that all we have are the dot products of x_i, x_j

If we take the derivative wrt a and set it equal to zero, we get the following solution, so we can solve for a_i:

$$\sum_{i=1}^{l} a_i y_i = 0$$

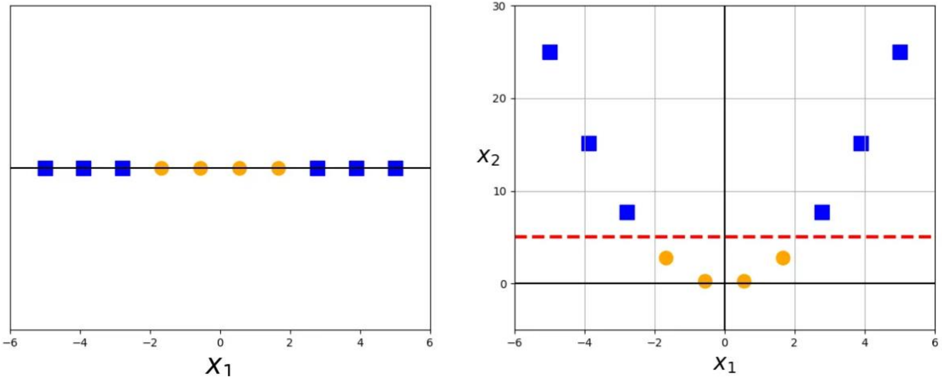
$$0 \leq a_i \leq C$$

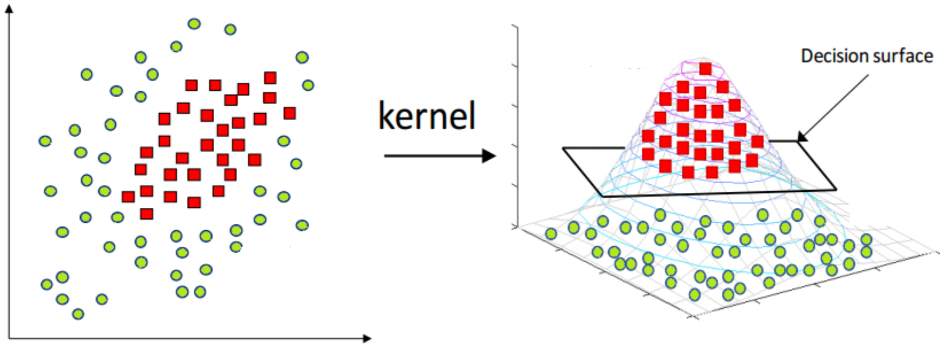
Now knowing the a_i we can find the weights **w** for the maximal margin separating hyperplane:

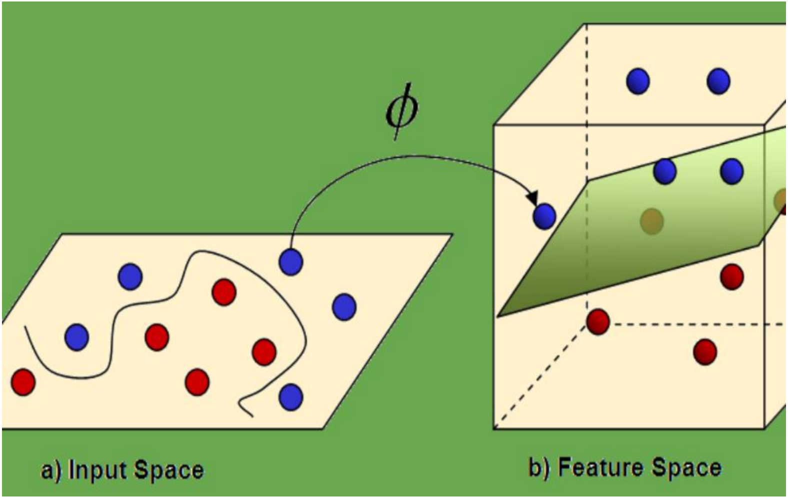
$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i$$

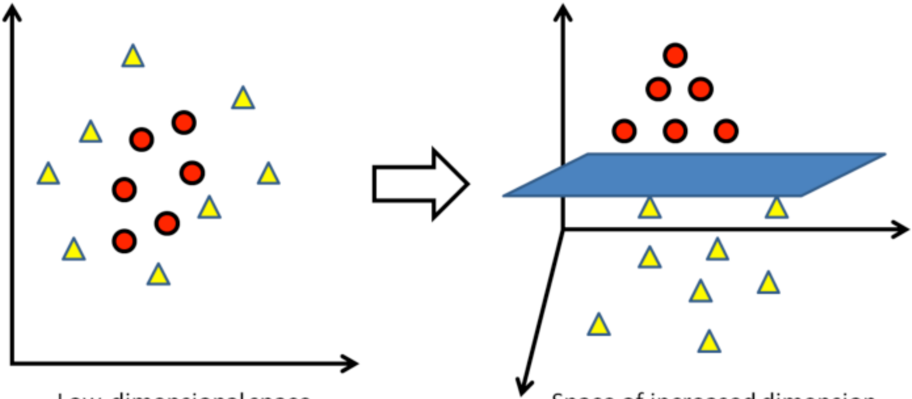
And now, after training and finding the **w** by this method, given an unknown point u measured on features x_i we can classify it by looking at the sign of:

$$f(x) = \mathbf{w} \bullet \mathbf{u} + b = \left(\sum_{i=1}^{l} a_i y_i \mathbf{x_i} \bullet \mathbf{u}\right) + b$$


−6
−4
−2
0
2
4
6
x_1
30
20
10
0
x_2
−6
−4
−2
0
2
4
6
x_1


kernel
Decision surface


ϕ
a) Input Space
b) Feature Space



Low-dimensional space

Space of increased dimension
after transformation

Non-linear SVMs

So, the function we end up optimizing is:

$L_d = \sum a_i - \frac{1}{2}\sum a_i a_j y_i y_j K(x_i \bullet x_j)$,

Kernel example: The polynomial kernel

$K(xi,xj) = (x_i \bullet x_j + 1)^p$