Key Ideas

- Two independent developments within last decade
  - New efficient separability of non-linear regions that use "kernel functions" : generalization of 'similarity' to new kinds of similarity measures based on dot products
  - Use of quadratic optimization problem to avoid 'local minimum' issues with neural nets
  - The resulting learning algorithm is an optimization algorithm rather than a greedy search

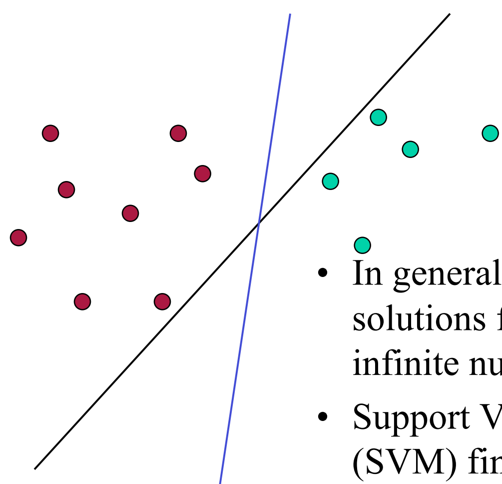
Organization

- Basic idea of support vector machines: just like 1-layer or multi-layer neural nets
  - Optimal hyperplane for linearly separable patterns
  - Extend to patterns that are not linearly separable by transformations of original data to map into new space – the Kernel function
- SVM algorithm for pattern recognition

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
- They have direct bearing on the optimum location of the decision surface
- We can show that the optimal hyperplane stems from the function class with the lowest "capacity"= # of independent features/parameters we can twiddle [note this is 'extra' material not covered in the lectures… you don't have to know this]

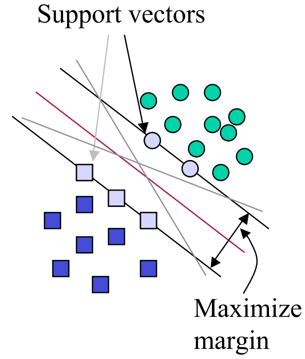
Recall from 1-layer nets : Which Separating Hyperplane?



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

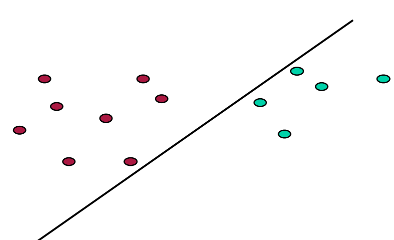
- Assume linear separability for now (we will relax this later)
- in 2 dimensions, can separate by a line
  - in higher dimensions, need hyperplanes

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 $\mathbf{w}$ (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



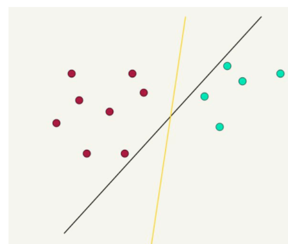
Find a,b,c, such that

$ax + by \geq c$ for red points

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

<u>Which</u> 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: <u>Which</u> points should influence optimality?
  - All points?
    - Linear regression
    - Neural nets
  - Or only "difficult points" close to decision boundary
    - Support vector machines

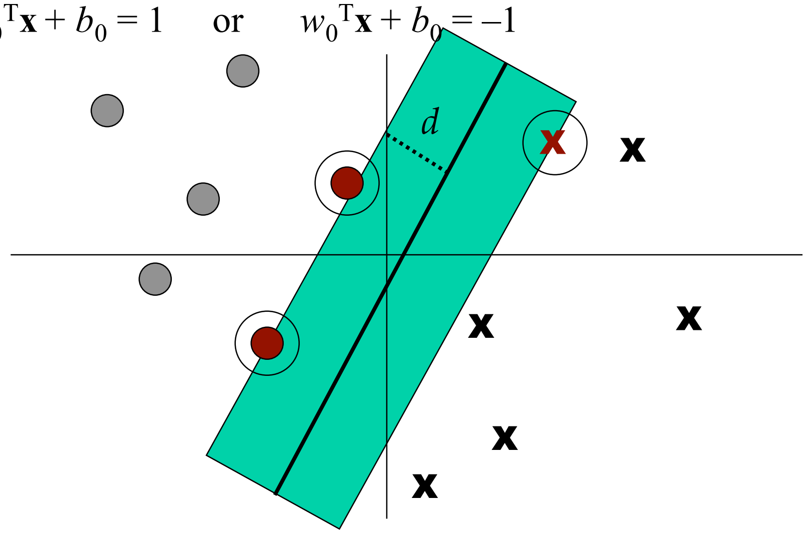


Support Vectors again for linearly separable case

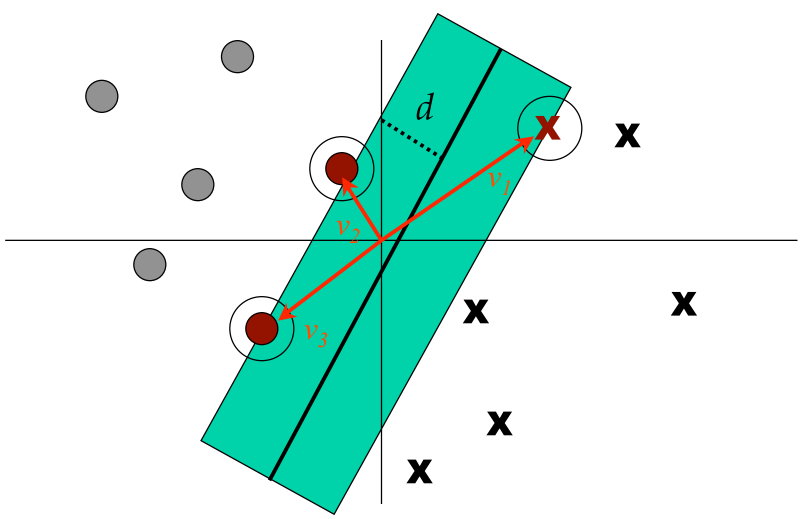
- Support vectors are the elements of the training set that would <u>change the position</u> of the dividing hyperplane if removed.
- Support vectors are the <u>critical</u> elements of the training set
- The problem of finding the optimal hyper plane is an optimization problem and can be solved by optimization 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 (or, rather, the 'tips' of the vectors

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

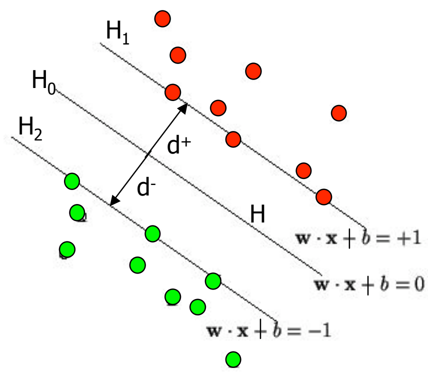
H_1 and H_2 are the planes:

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 <u>Support Vectors</u>

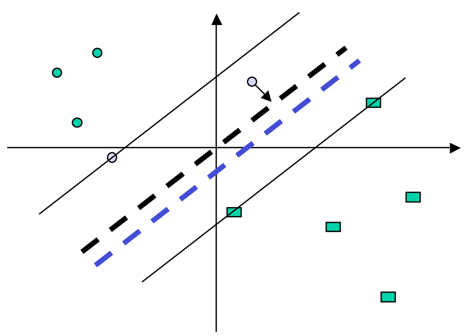
The plane H_0 is the median in between, where $w \bullet x_i + b = 0$



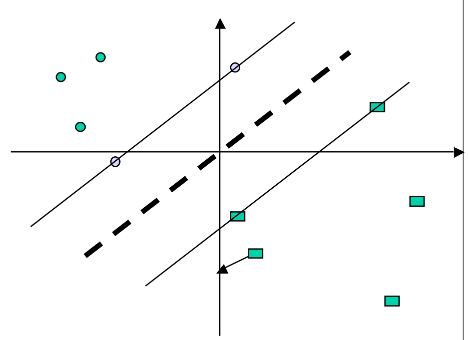
d+ = the shortest distance to the closest positive point

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

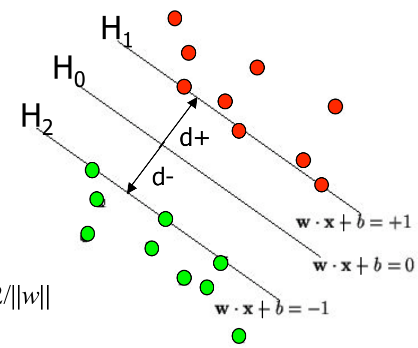
$$\mathbf{w}^T\mathbf{x} + b \geq 0 \text{ for } d_i = +1$$
$$\mathbf{w}^T\mathbf{x} + b < 0 \text{ for } d_i = -1$$

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: $\mathbf{y_i(x_i\bullet w) \geq 1}$**

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: $\frac{1}{2}\,||w||^2$ (Note this is a quadratic function)
  s.t. g: $y_i(\mathbf{w}\bullet\mathbf{x}_i)-\mathbf{b} = \mathbf{1}$ or $[y_i(\mathbf{w}\bullet\mathbf{x}_i)-\mathbf{b}] - \mathbf{1} = \mathbf{0}$

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 (thus avoiding a problem we had with neural nets!)