10-601
Machine Learning

Support Vector Machine

Types of classifiers

- We can divide the large variety of classification approaches into roughly three major types

  1. Instance based classifiers
     - Use observation directly (no models)
     - e.g. K nearest neighbors

  2. Generative:
     - build a generative statistical model
     - e.g., Bayesian networks

  3. Discriminative
     - directly estimate a decision rule/boundary
     - e.g., decision tree

Ranking classifiers

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774
DT	–	.647	.639	.824	.843	.762	.777	.562	.607	.708	.763
DT	PLT	.651	.618	.824	.843	.762	.777	.575	.594	.706	.761
LR	–	.636	.545	.823	.852	.743	.734	.620	.645	.700	.710
LR	ISO	.627	.567	.818	.847	.735	.742	.608	.589	.692	.703
LR	PLT	.630	.500	.823	.852	.743	.734	.593	.604	.685	.695
NB	ISO	.579	.468	.779	.820	.727	.733	.572	.555	.654	.661
NB	PLT	.576	.448	.780	.824	.738	.735	.537	.559	.650	.654
NB	–	.496	.562	.781	.825	.738	.735	.347	-.633	.481	.489

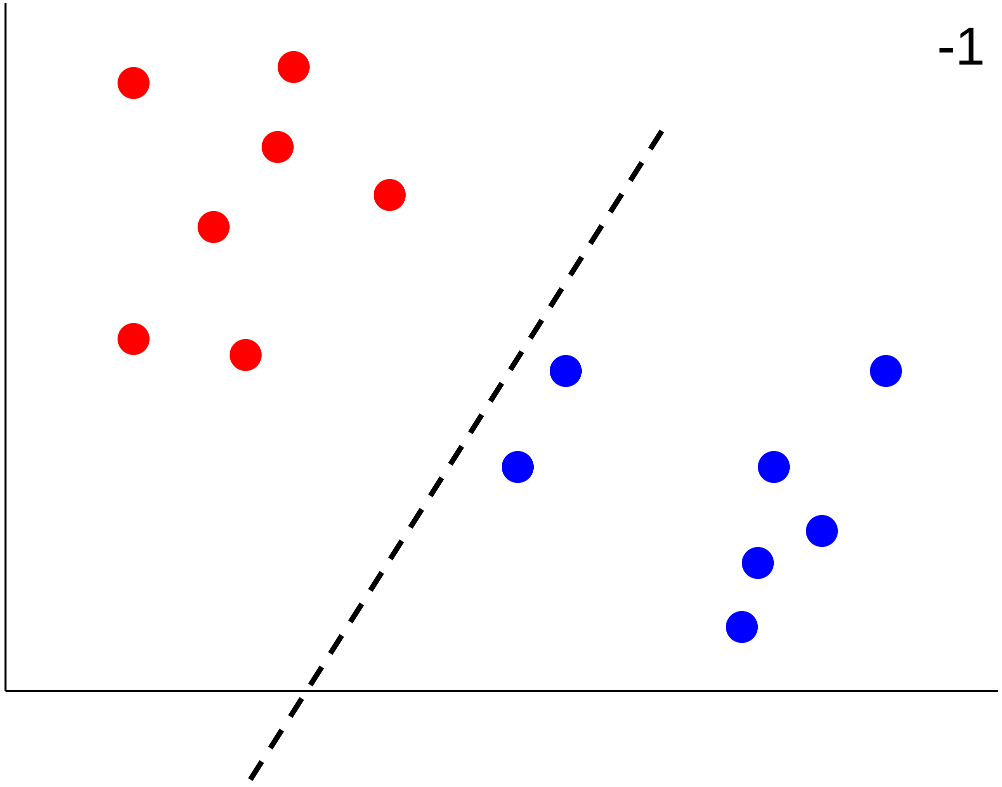
Rich Caruana & Alexandru Niculescu-Mizil, An Empirical Comparison of Supervised Learning Algorithms, ICML 2006

Regression classifiers

Recall our regression classifiers

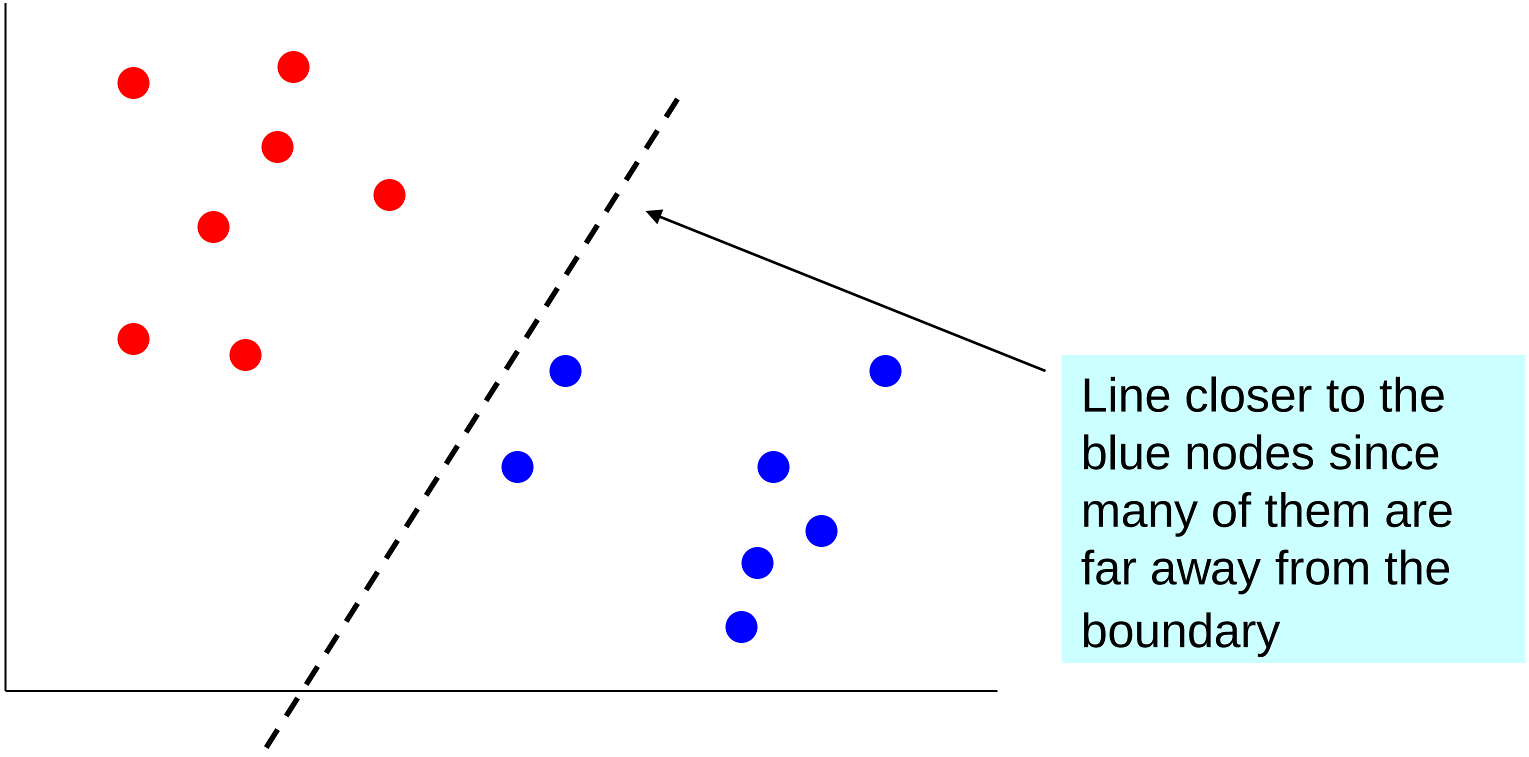
+1 if $\mathrm{sign}(w^T x+b) \geq 0$

-1 if $\mathrm{sign}(w^T x+b) < 0$



Regression classifiers

Recall our regression classifiers

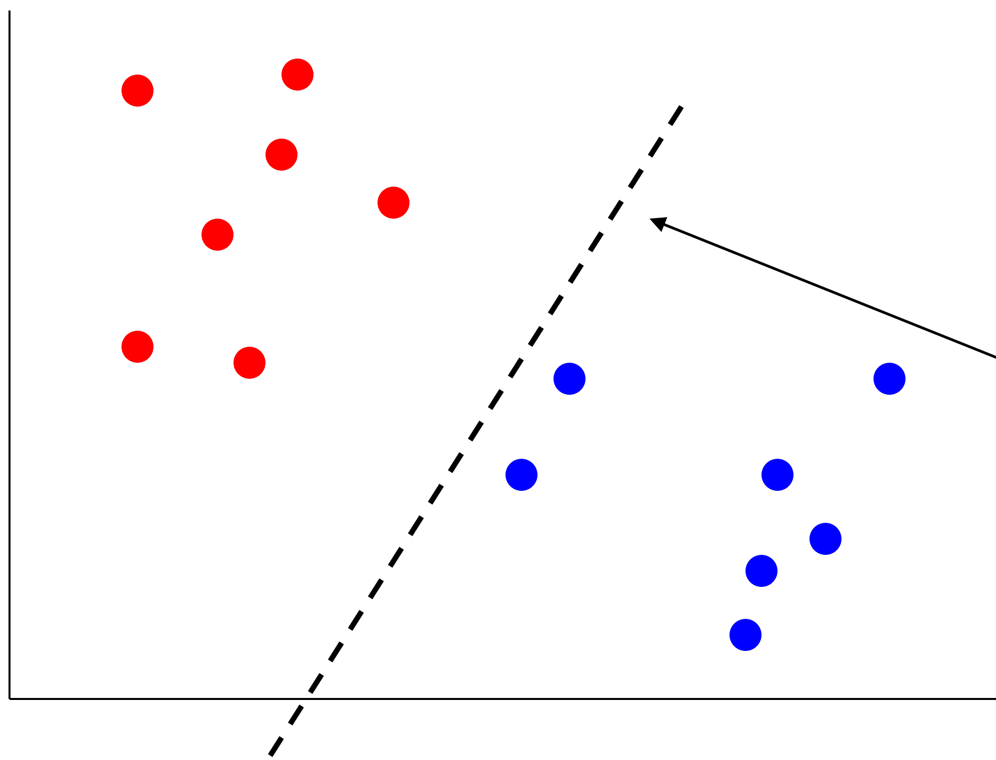


Regression classifiers

Recall our regression classifiers

$$\min_w \sum_i (y_i - \mathrm{w}^{\mathrm{T}} x_i)^2$$

Goes over all points x (even for logistic regression)



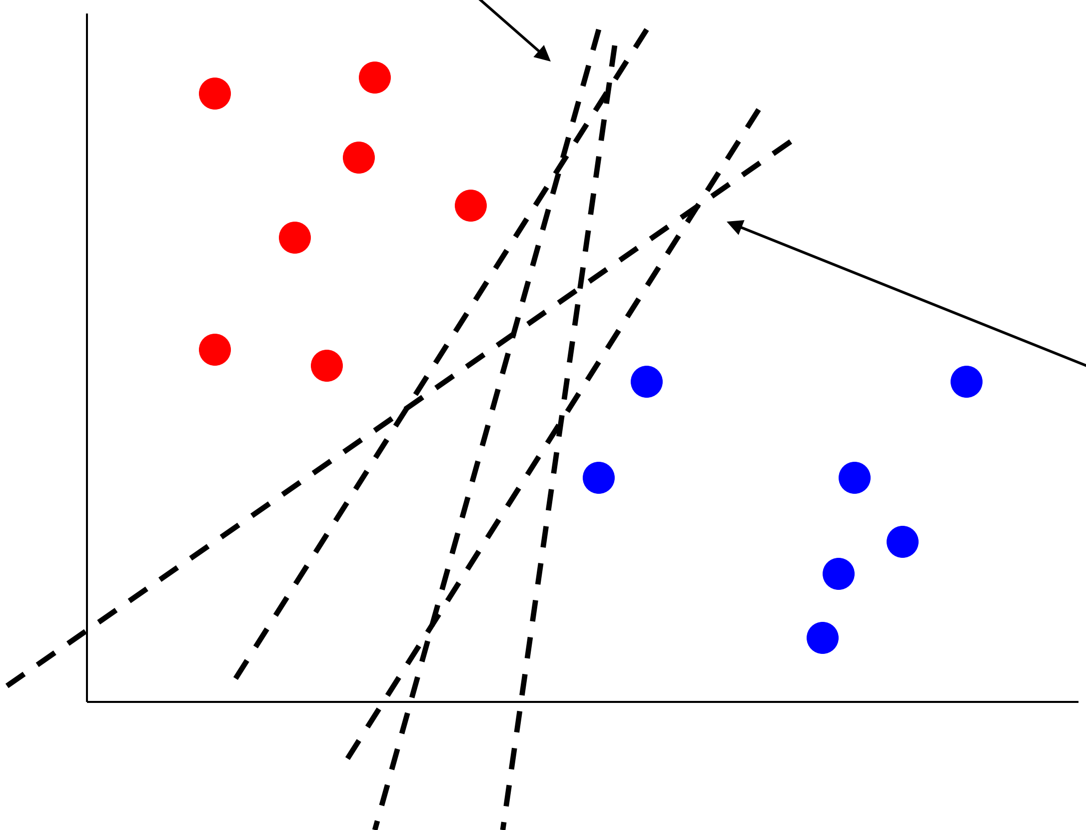
Line closer to the blue nodes since many of them are far away from the boundary

Regression classifiers

Recall our regression classifiers

$$\min_w \sum_i (y_i - \mathrm{w}^T x_i)^2$$

Many more possible classifiers

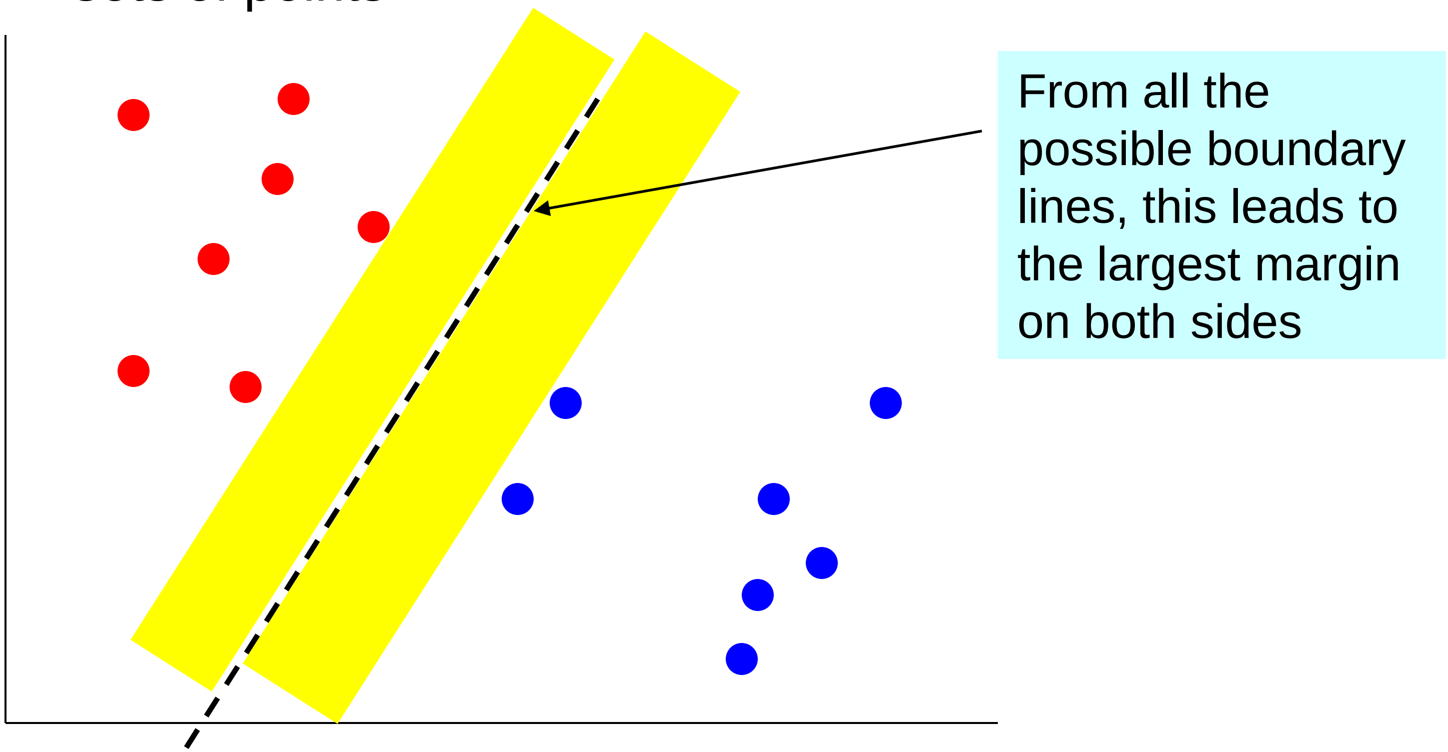


Goes over all points x (even in LR settings)

Line closer to the blue nodes since many of them are far away from the boundary

Max margin classifiers

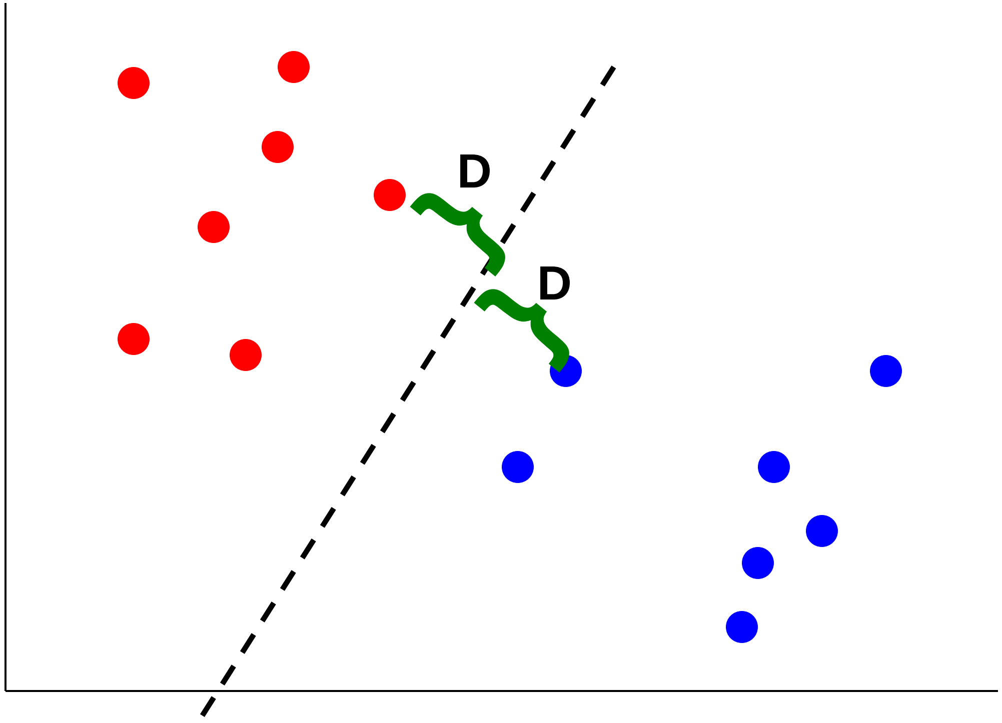
- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest **margin** from both sets of points



Max margin classifiers

• Instead of fitting all points, focus on boundary points

• Learn a boundary that leads to the largest margin from points on both sides



Why?

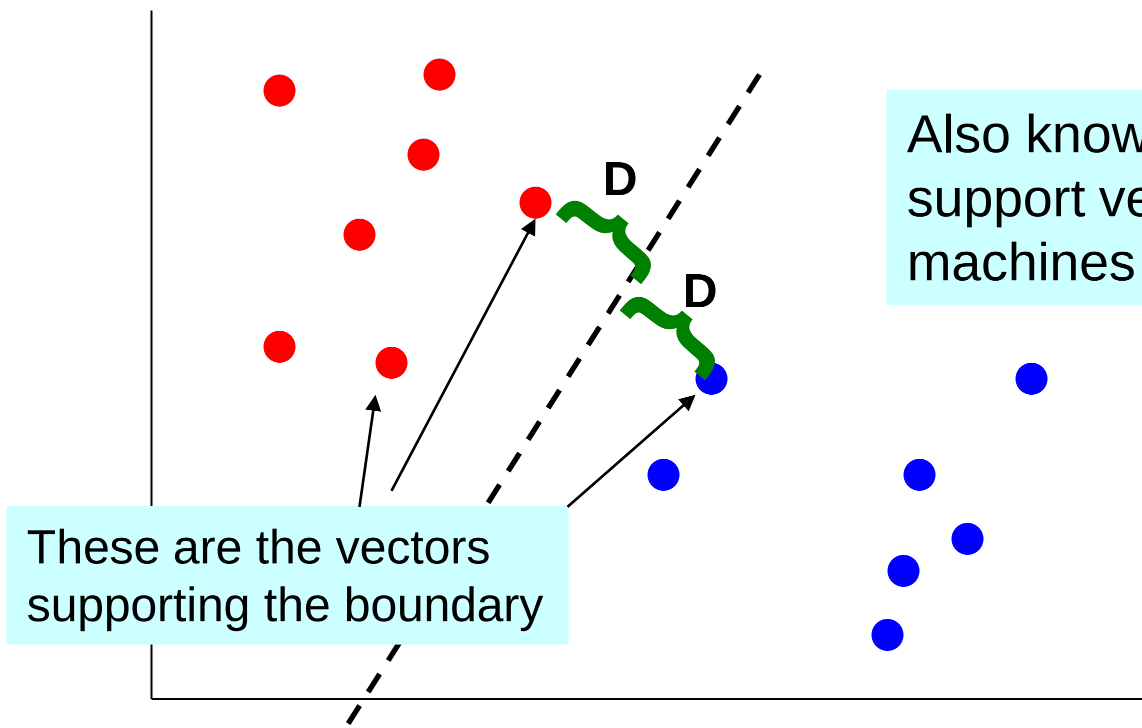
• Intuitive, 'makes sense'

• Some theoretical support

• Works well in practice

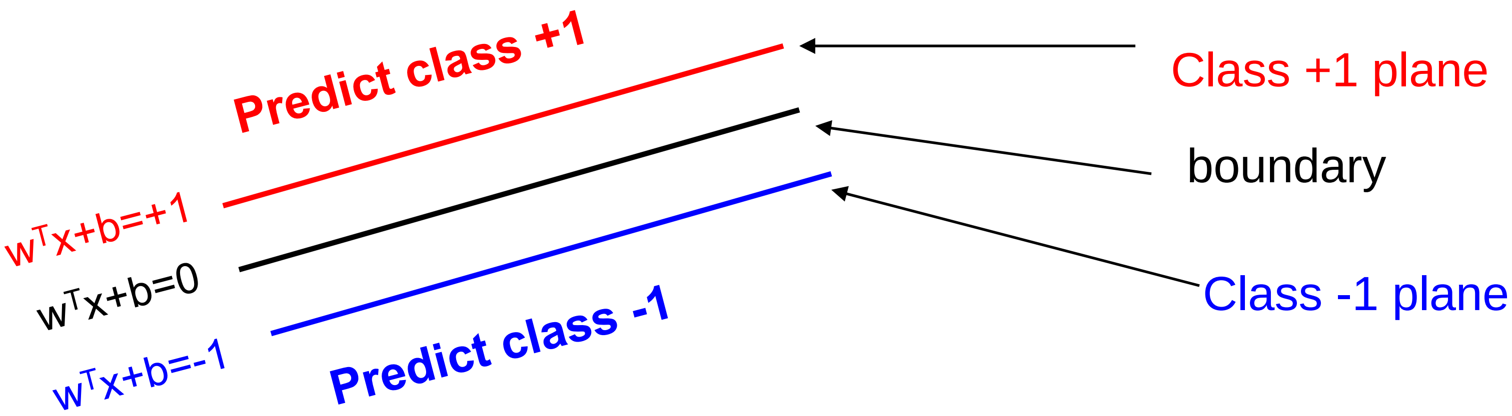
Max margin classifiers

- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from points on both sides



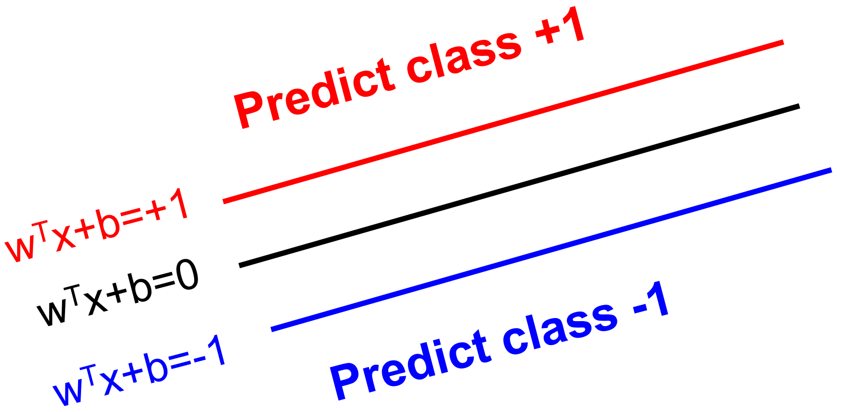
Also known as linear support vector machines (SVMs)

Specifying a max margin classifier



Classify as +1	if	$w^Tx+b \geq 1$
Classify as -1	if	$w^Tx+b \leq -1$
Undefined	if	$-1 < w^Tx+b < 1$

Specifying a max margin classifier

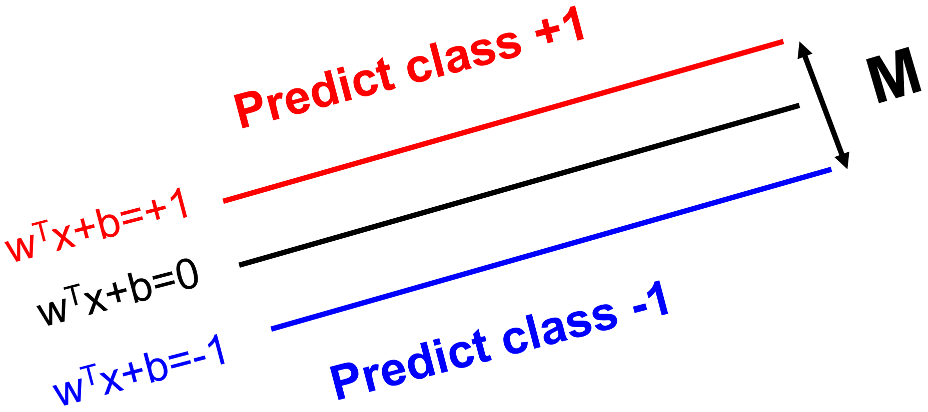


Is the linear separation assumption realistic?

We will deal with this shortly, but lets assume it for now

Classify as +1	if	$w^Tx+b \geq 1$
Classify as -1	if	$w^Tx+b \leq -1$
Undefined	if	$-1 < w^Tx+b < 1$

Maximizing the margin



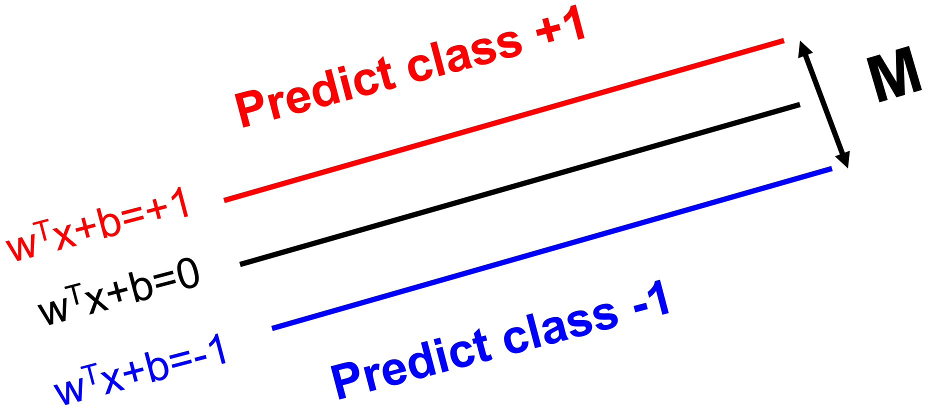
Classify as +1 if $w^Tx+b \geq 1$

Classify as -1 if $w^Tx+b \leq -1$

Undefined if $-1 < w^Tx+b < 1$

- Lets define the width of the margin by M
- How can we encode our goal of maximizing M in terms of our parameters (w and b)?
- Lets start with a few obsevrations

Maximizing the margin



Classify as +1 if $w^Tx+b \geq 1$

Classify as -1 if $w^Tx+b \leq -1$

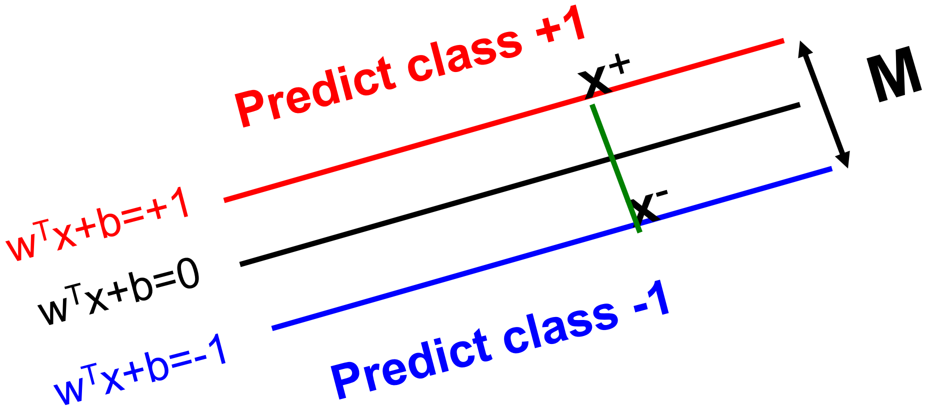
Undefined if $-1 < w^Tx+b < 1$

- Observation 1: the vector w is orthogonal to the +1 plane
- Why?

Let u and v be two points on the +1 plane, then for the vector defined by u and v we have $w^T(u-v) = 0$

Corollary: the vector w is orthogonal to the -1 plane

Maximizing the margin



Classify as +1 if $w^T x+b \geq 1$

Classify as -1 if $w^T x+b \leq -1$

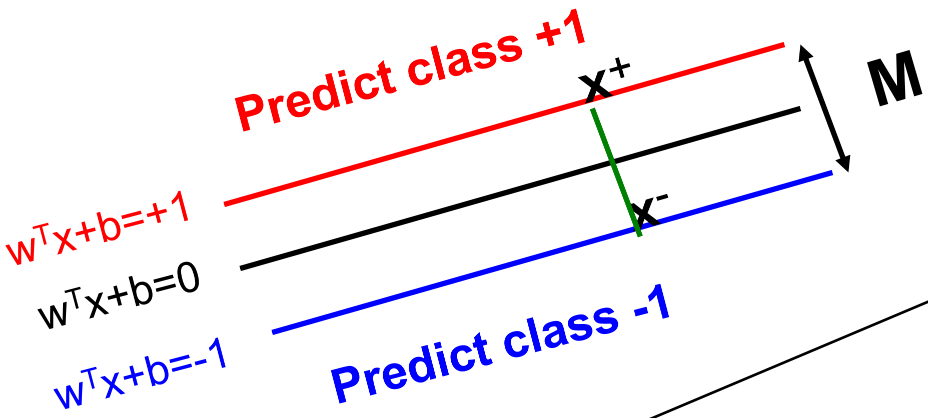
Undefined if $-1 < w^T x+b < 1$

- Observation 1: the vector w is orthogonal to the +1 and -1 planes
- Observation 2: if x^+ is a point on the +1 plane and x^- is the closest point to x^+ on the -1 plane then

$$x^+ = \lambda w + x^-$$

Since w is orthogonal to both planes we need to 'travel' some distance along w to get from x^+ to x^-

Putting it together



- $w^T x^+ + b = +1$
- $w^T x^- + b = -1$
- $x^+ = \lambda w + x^-$
- $| x^+ - x^- | = M$

$w^T x^+ + b = +1$

$\Rightarrow$

$w^T (\lambda w + x^-) + b = +1$

$\Rightarrow$

$w^Tx^- + b \; + \lambda w^Tw = +1$

$\Rightarrow$

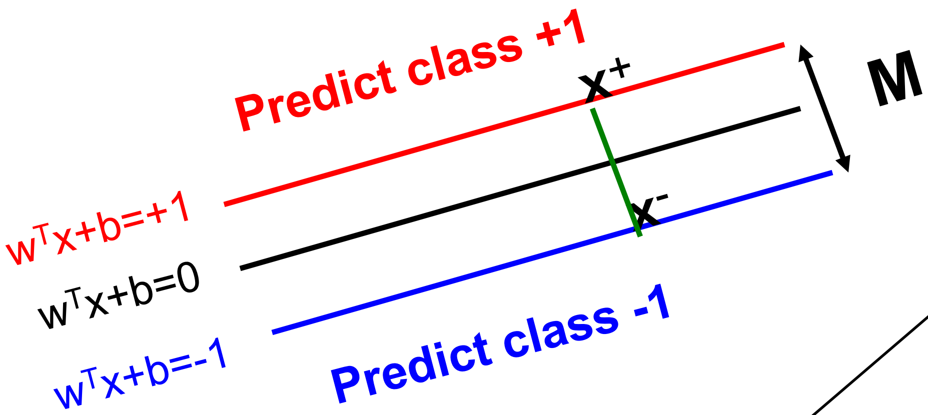
$-1 \; + \lambda w^Tw = +1$

$\Rightarrow$

$\lambda = 2/w^Tw$

We can now define M in terms of w and b

Putting it together



- $w^T x^+ + b = +1$
- $w^T x^- + b = -1$
- $x^+ = \lambda w + x^-$
- $| x^+ - x^- | = M$
- $\lambda = 2/w^Tw$

$M = |x^+ - x^-|$

$\Rightarrow$

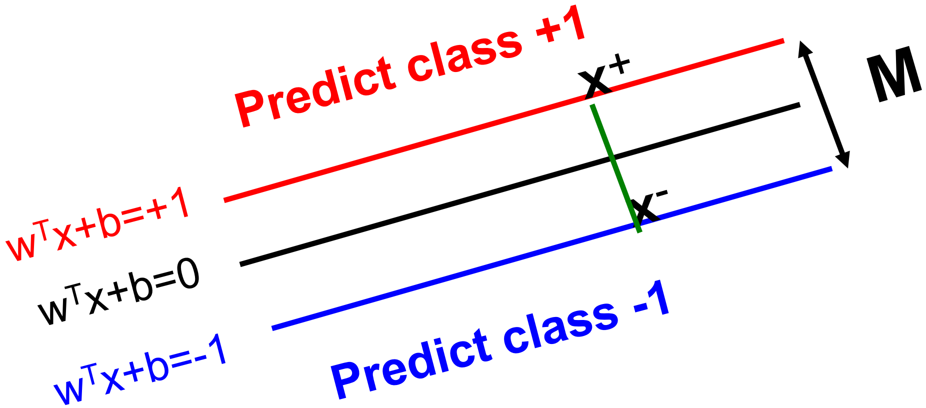
$$M = |\lambda w| = \lambda |w| = \lambda \sqrt{w^T w}$$

$\Rightarrow$

$$M = 2\frac{\sqrt{w^T w}}{w^T w} = \frac{2}{\sqrt{w^T w}}$$

We can now define M in terms of w and b

Finding the optimal parameters



$$M = \frac{2}{\sqrt{w^T w}}$$

We can now search for the optimal parameters by finding a solution that:

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Several optimization methods can be used: Gradient descent, simulated annealing, EM etc.

Quadratic programming (QP)

Quadratic programming solves optimization problems of the following form:

$$\min_U \frac{u^T R u}{2} + d^T u + c$$

u -vector (unknown)
R – diagonal matrix
d – vector
c - scalar

Quadratic term

subject to n inequality constraints:

$$a_{11}u_1 + a_{12}u_2 + \ldots \leq b_1$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$a_{n1}u_1 + a_{n2}u_2 + \ldots \leq b_n$$

and k equivalency constraints:

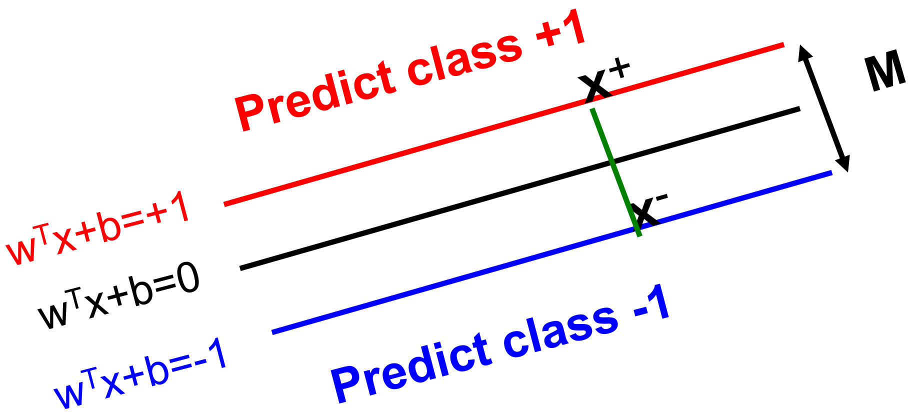
$$a_{n+1,1}u_1 + a_{n+1,2}u_2 + \ldots = b_{n+1}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$a_{n+k,1}u_1 + a_{n+k,2}u_2 + \ldots = b_{n+k}$$

When a problem can be specified as a QP problem we can use solvers that are better than gradient descent or simulated annealing

SVM as a QP problem



$$M = \frac{2}{\sqrt{w^T w}}$$

$$\min_U \frac{u^T R u}{2} + d^T u + c$$

subject to n inequality constraints:

$$a_{11}u_1 + a_{12}u_2 + \ldots \le b_1$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$a_{n1}u_1 + a_{n2}u_2 + \ldots \le b_n$$

and k equivalency constraints:

$$a_{n+1,1}u_1 + a_{n+1,2}u_2 + \ldots = b_{n+1}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$a_{n+k,1}u_1 + a_{n+k,2}u_2 + \ldots = b_{n+k}$$

Min $(w^T w)/2$

subject to the following inequality constraints:

For all x in class + 1

$w^T x+b \ge 1$

For all x in class - 1

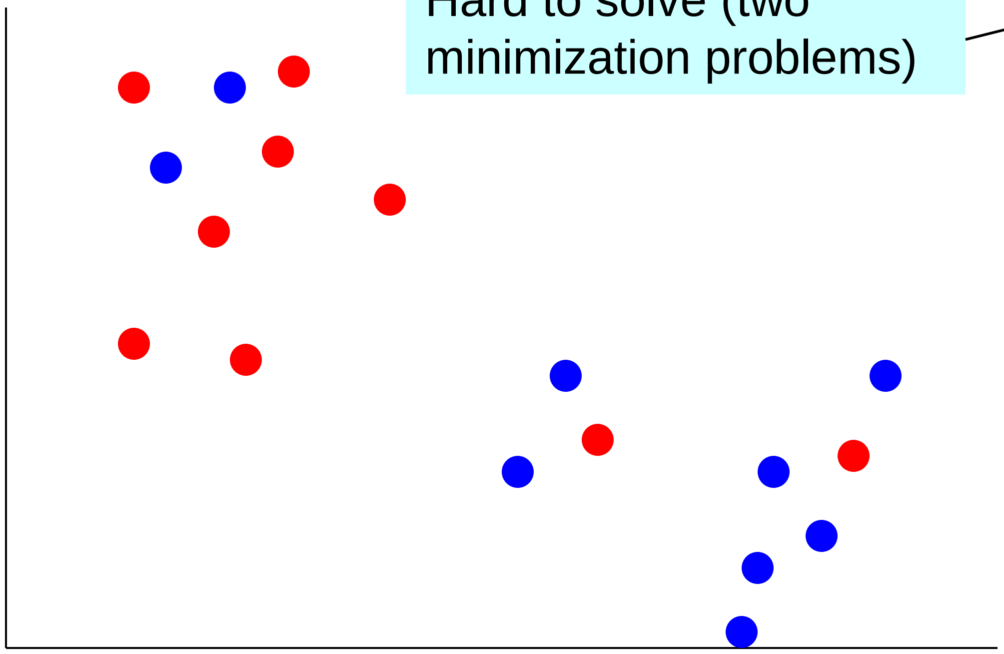
$w^T x+b \le -1$

A total of n constraints if we have n input samples

Non linearly separable case

- So far we assumed that a linear plane can perfectly separate the points
- But this is not usally the case

 - noise, outliers



Hard to solve (two minimization problems)

How can we convert this to a QP problem?

- Minimize training errors?

$\min w^T w$

min #errors

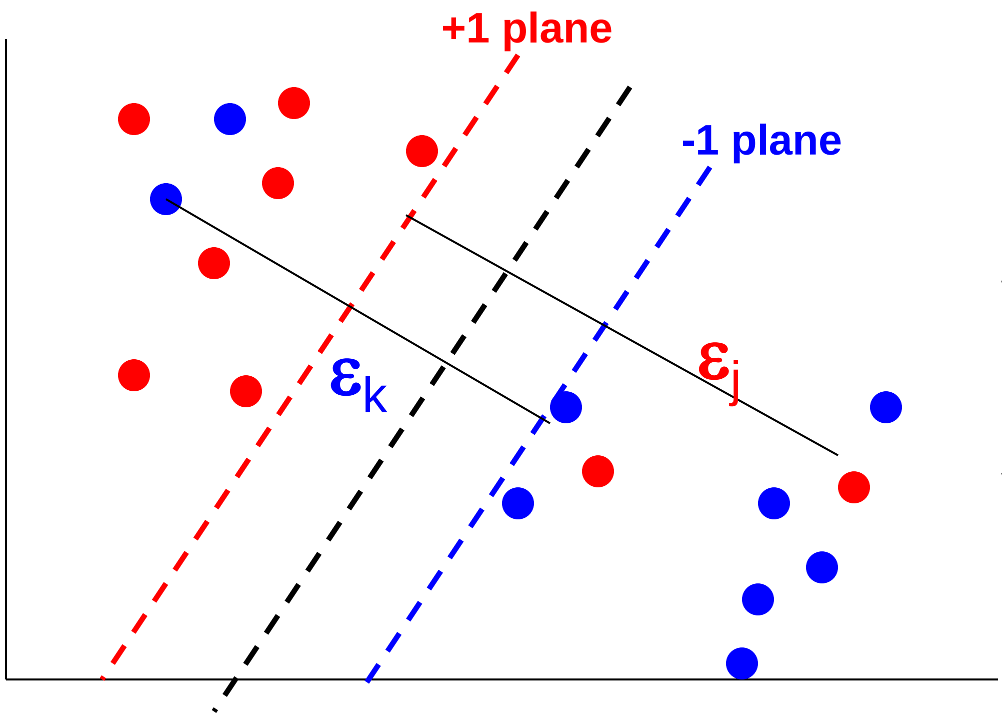
- Penalize training errors:

$\min w^T w + C^*(\#\text{errors})$

Hard to encode in a QP problem

Non linearly separable case

- Instead of minimizing the number of misclassified points we can minimize the *distance* between these points and their correct plane



The new optimization problem is:

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

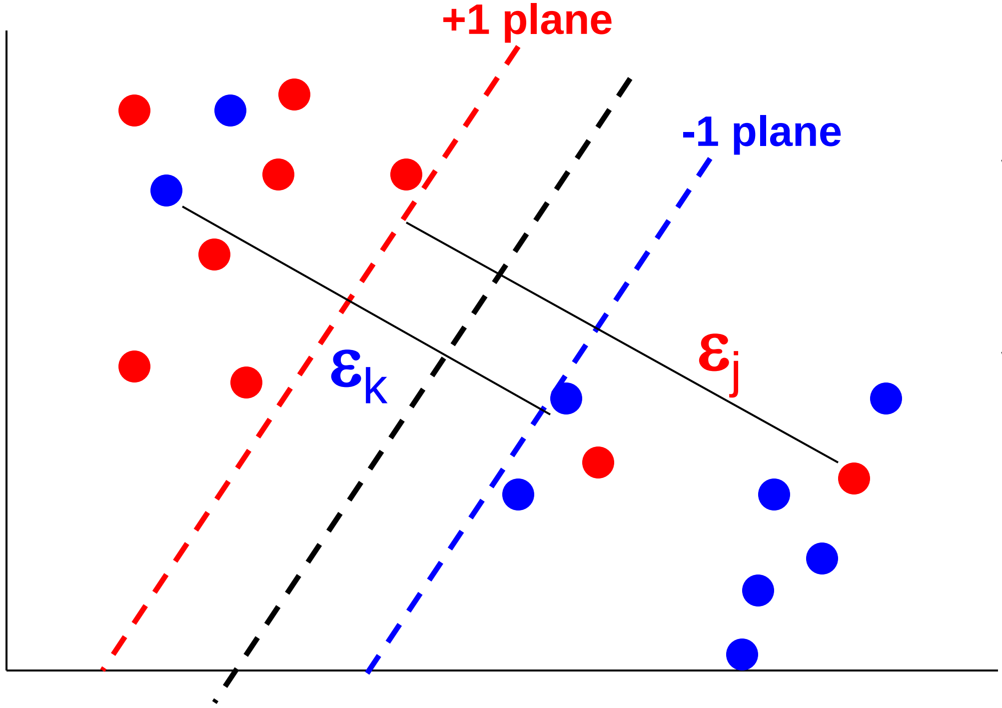
$w^T x + b \geq 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x + b \leq -1 + \varepsilon_i$

Wait. Are we missing something?

Final optimization for non linearly separable case



The new optimization problem is:

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

$w^T x + b \geq 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x + b \leq -1 + \varepsilon_i$

A total of n constraints

For all i

$\varepsilon_I \geq 0$

Another n constraints

Where we are

Two optimization problems: For the separable and non separable cases

$$\min_w \frac{w^T w}{2}$$

For all x in class + 1

$w^T x+b \geq 1$

For all x in class - 1

$w^T x+b \leq -1$

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

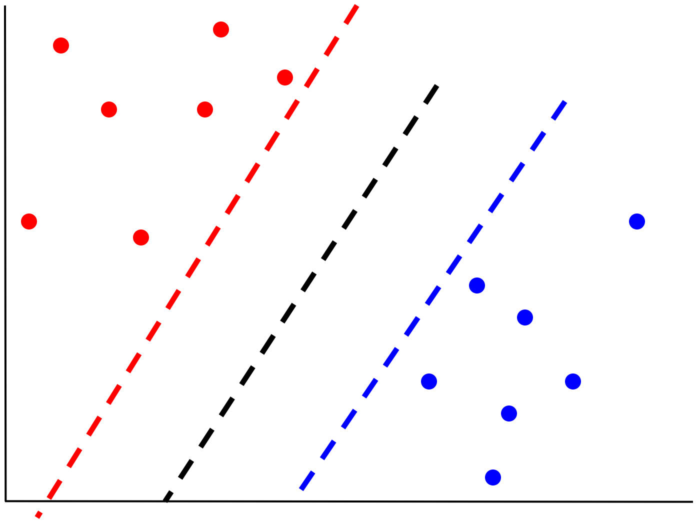
$w^T x+b \geq 1- \boldsymbol{\varepsilon}_i$

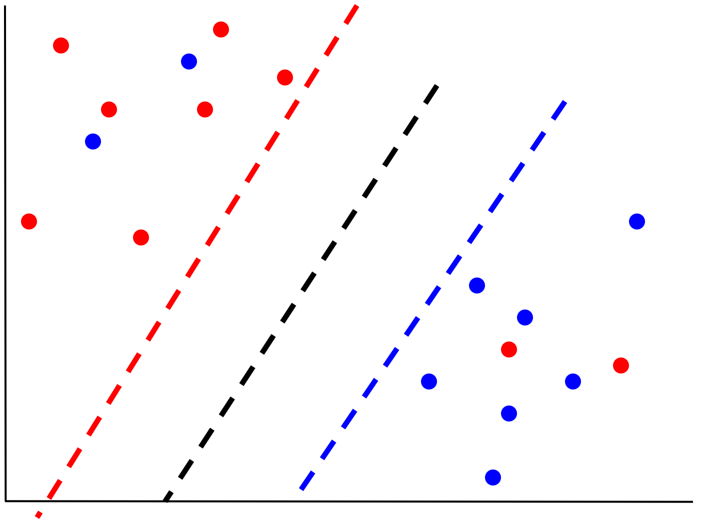
For all x_i in class - 1

$w^T x+b \leq -1+ \boldsymbol{\varepsilon}_i$

For all i

$\boldsymbol{\varepsilon}_I \geq 0$





Where we are

Two optimization problems: For the separable and non separable cases

Min $(w^T w)/2$

For all x in class + 1

$w^T x + b \geq 1$

For all x in class - 1

$w^T x + b \leq -1$

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C \varepsilon_i$$

For all x_i in class + 1

$w^T x + b \geq 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x + b \leq -1 + \varepsilon_i$

For all i

$\varepsilon_I \geq 0$

- Instead of solving these QPs directly we will solve a dual formulation of the SVM optimization problem
- The main reason for switching to this type of representation is that it would allow us to use a neat trick that will make our lives easier (and the run time faster)

An alternative (dual) representation of the SVM QP

- We will start with the linearly separable case
- Instead of encoding the correct classification rule and constraint we will use LaGrange multiplies to encode it as part of the our minimization problem

Min $(w^T w)/2$

For all x in class +1

$w^T x + b \geq 1$

For all x in class -1

$w^T x + b \leq -1$

Why? $\Downarrow$

Min $(w^T w)/2$

$(w^T x_i + b) y_i \geq 1$

An alternative (dual) representation of the SVM QP

Min $(w^Tw)/2$

$(w^Tx_i+b)y_i \geq 1$

- We will start with the linearly separable case
- Instead of encoding the correct classification rule a constraint we will use Lagrange multiplies to encode it as part of the our minimization problem

Recall that Lagrange multipliers can be applied to turn the following problem:

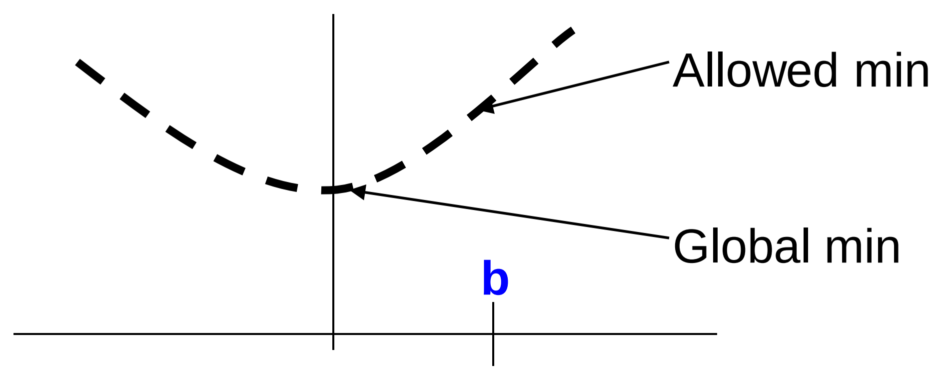
$\min_x x^2$

s.t. $x \geq b$

To

$\min_x \max_\alpha x^2 - \alpha(x-b)$

s.t. $\alpha \geq 0$



Lagrange multiplier for SVMs

Dual formulation

$$\min_{w,b} \max_{\alpha} \frac{\mathrm{w}^{\mathrm{T}}\mathrm{w}}{2} - \sum_i \alpha_i [(\mathrm{w}^{\mathrm{T}} x_i + b) y_i - 1]$$

$$\alpha_i \geq 0 \qquad \forall i$$

Original formulation

Min $(w^T w)/2$

$(w^T x_i + b) y_i \geq 1$

Using this new formulation we can derive w and b by taking the derivative w.r.t. w and α leading to:

$$w = \sum_i \alpha_i x_i y_i$$

$$b = y_i - \mathrm{w}^{\mathrm{T}} x_i$$

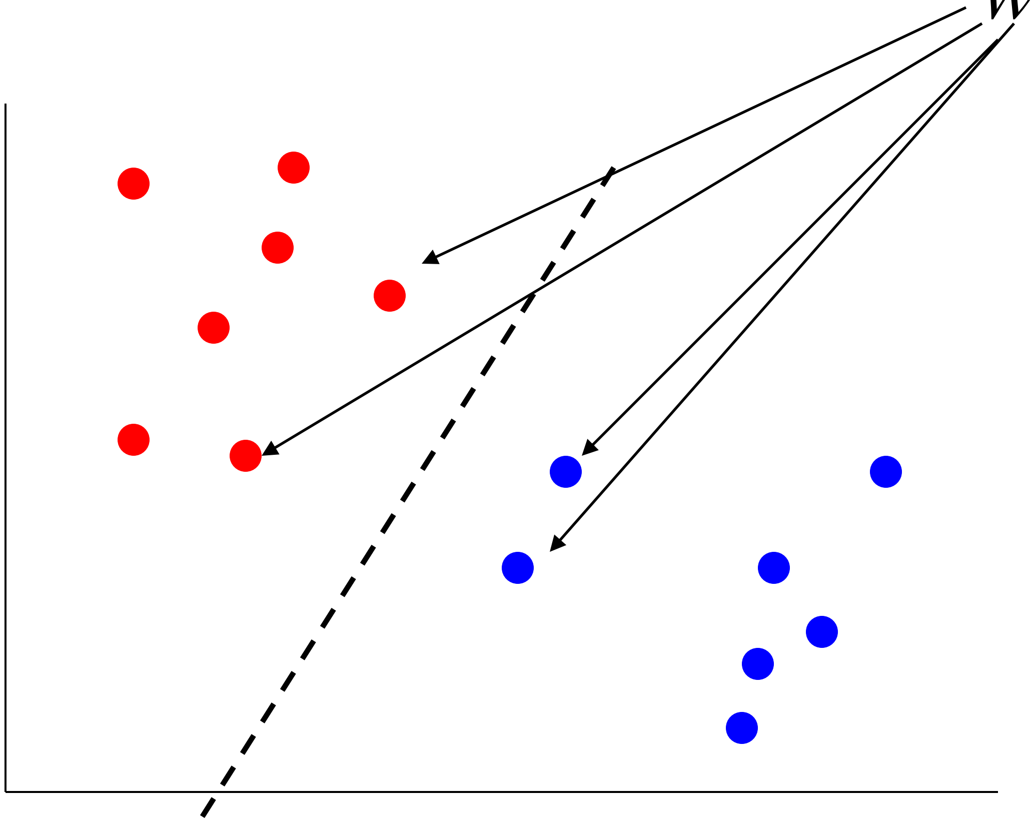
$$for \quad i \quad s.t. \qquad \alpha_i > 0$$

Finally, taking the derivative w.r.t. b we get:

$$\sum_i \alpha_i y_i = 0$$

Dual SVM - interpretation

$$w = \sum_i \alpha_i x_i y_i$$



For α's that are not 0

Dual SVM for linearly separable case

Substituting w into our target function and using the additional constraint we get:

$$\min_{w,b} \frac{\mathrm{w}^{\mathrm{T}}\mathrm{w}}{2} - \sum_i \alpha_i [(\mathrm{w}^{\mathrm{T}} x_i + b) y_i - 1]$$

$$\alpha_i \geq 0 \qquad \forall i$$

$$w = \sum_i \alpha_i x_i y_i$$

$$b = y_i - \mathrm{w}^{\mathrm{T}} x_i$$

$$for \quad i \quad s.t. \quad \alpha_i > 0$$

$$\sum_i \alpha_i y_i = 0$$

Dual formulation

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i} \mathbf{x_j}$$

$$\sum_i \alpha_i \mathrm{y_i} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dual SVM for linearly separable case

Our dual target function: $$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x}_{\mathbf{i}} \mathbf{x}_{\mathbf{j}}$$

$$\sum_{i} \alpha_i \mathrm{y}_{\mathrm{i}} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dot product for all training samples

Dot product with training samples

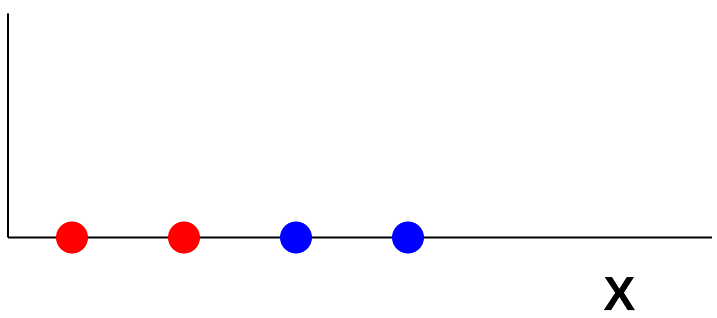
To evaluate a new sample x_j we need to compute:

$$\mathrm{w}^{\mathrm{T}} x_j + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y}_{\mathrm{i}} \mathbf{x}_{\mathbf{i}} \mathbf{x}_{\mathbf{j}} + b$$

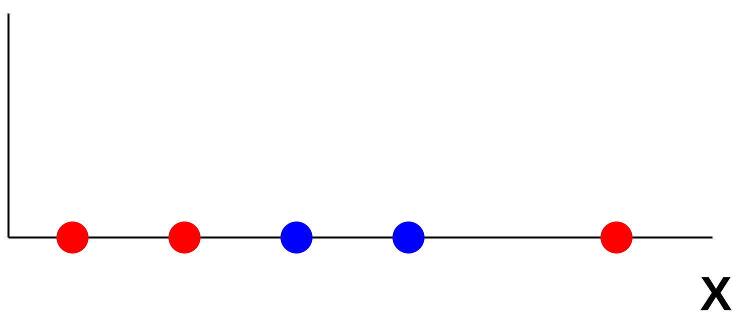
Is this too much computational work (for example when using transformation of the data)?

Classifying in 1-d

Can an SVM correctly classify this data?

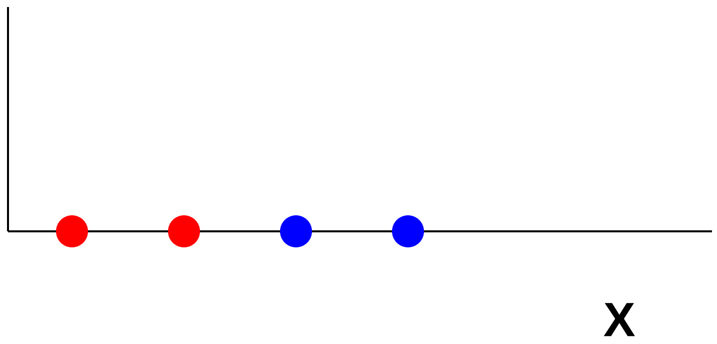


What about this?

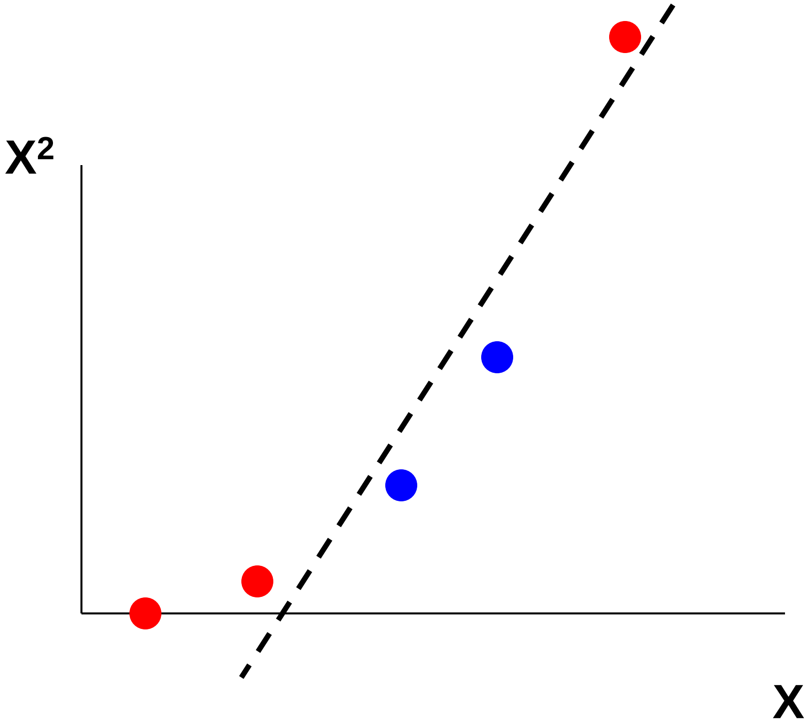


Classifying in 1-d

Can an SVM correctly classify this data?



And now?

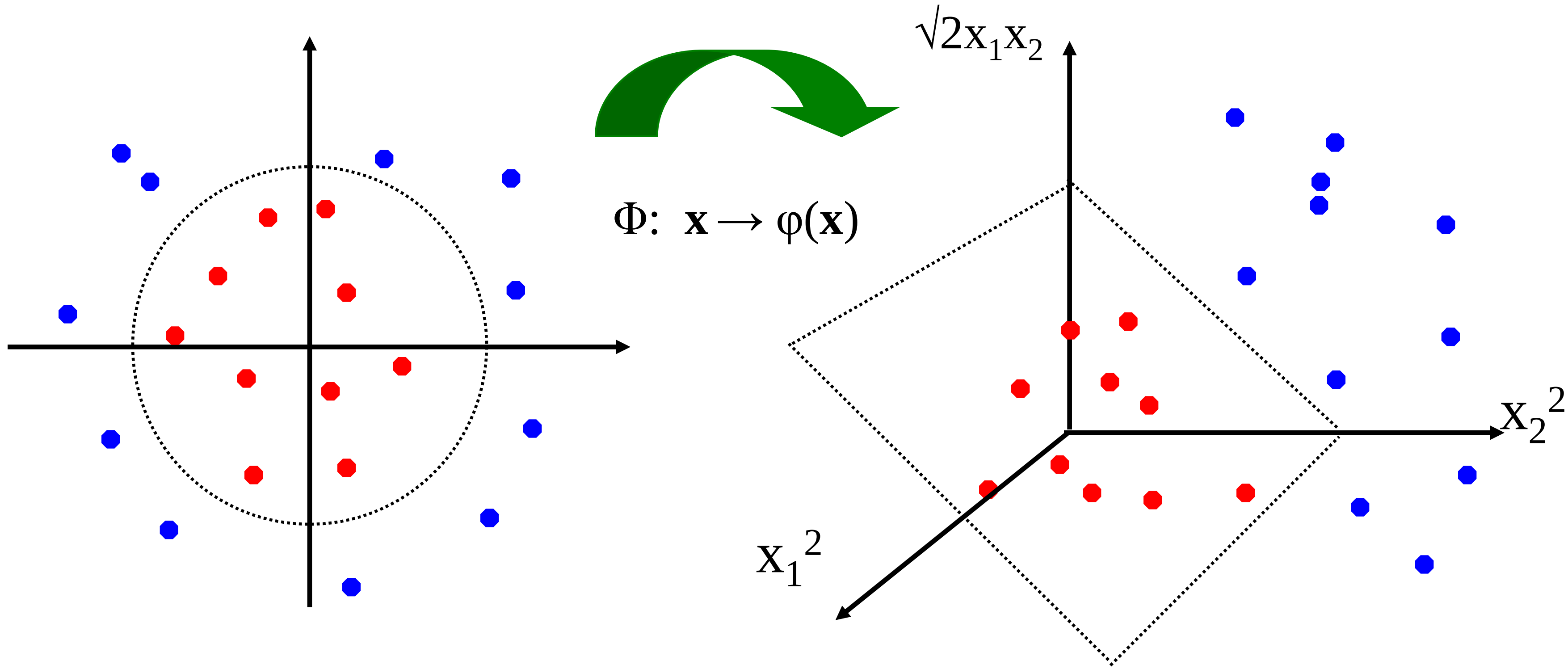


Non-linear SVMs: 2D

- The original input space (x) can be mapped to some higher-dimensional feature space ($\varphi(\mathbf{x})$)where the training set is separable:

$x=(x_1,x_2)$

$\varphi(\mathbf{x}) =(x_1^2,x_2^2,\sqrt{2}x_1x_2)$

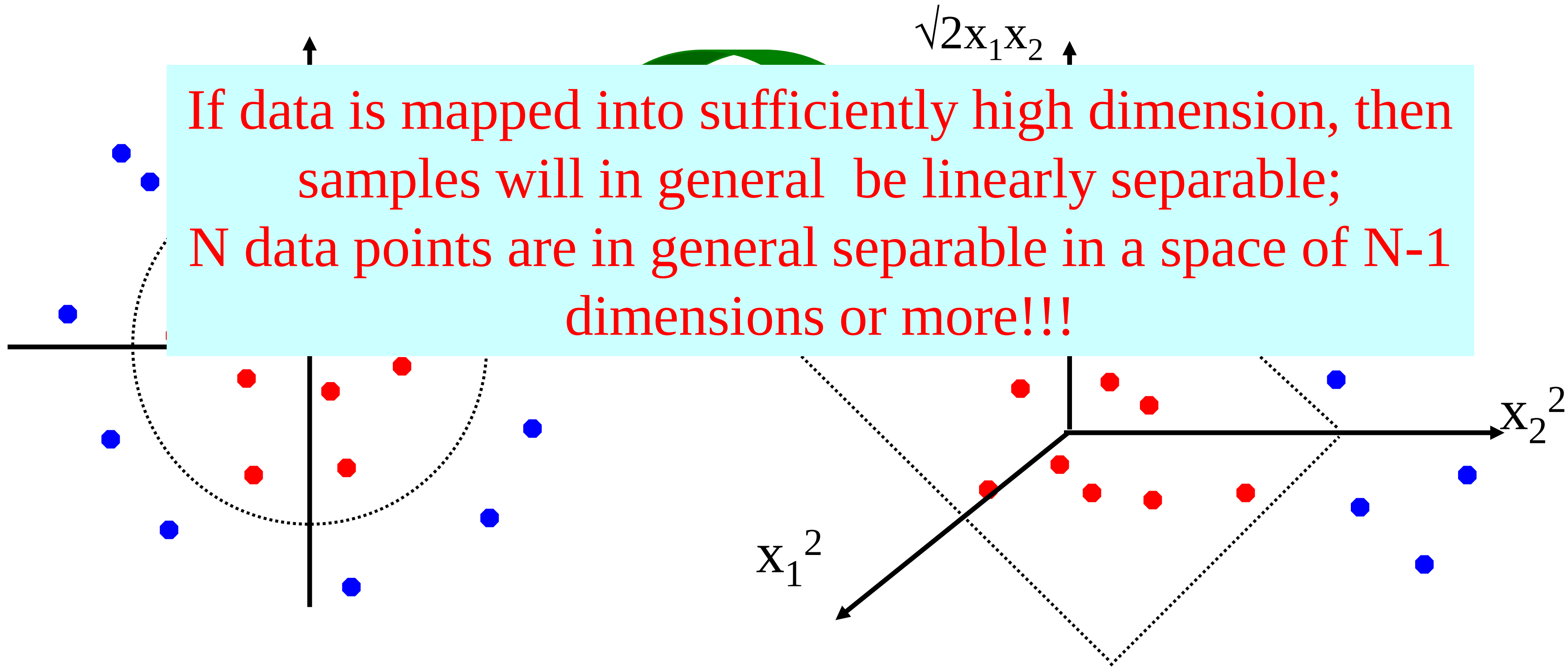


This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Non-linear SVMs: 2D

- The original input space (x) can be mapped to some higher-dimensional feature space ($\varphi(\mathbf{x})$)where the training set is separable:

$x=(x_1,x_2)$ $\qquad\qquad$ $\varphi(\mathbf{x}) =(x_1^2,x_2^2,\sqrt{2}x_1x_2)$

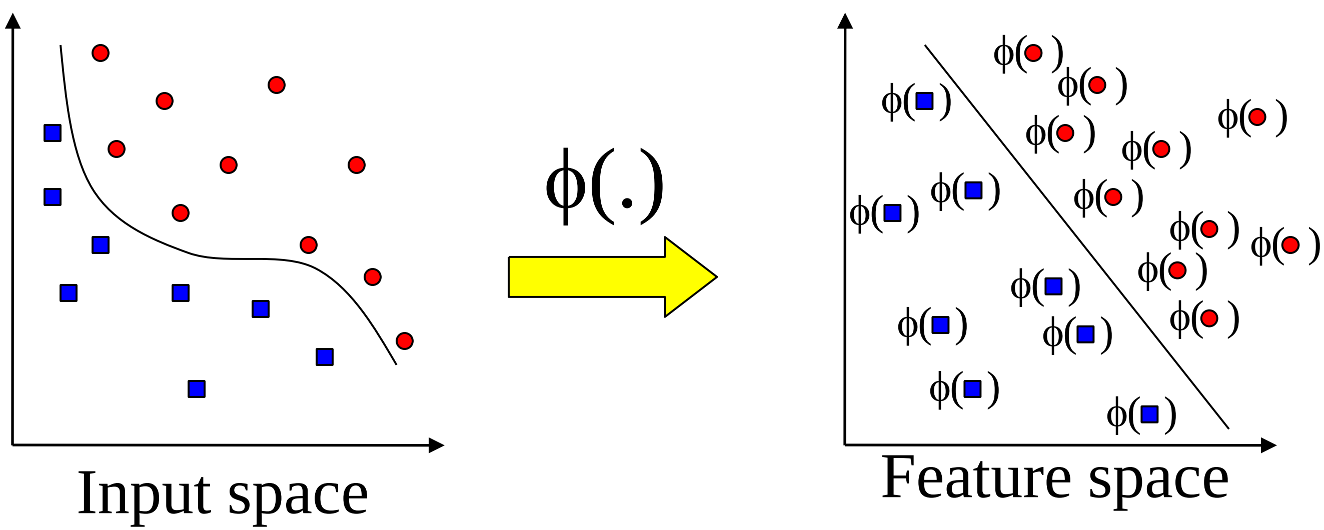


If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Transformation of Inputs

- Possible problems
  - High computation burden due to high-dimensionality
  - Many more parameters
- SVM solves these two issues simultaneously
  - “Kernel tricks” for efficient computation
  - Dual formulation only assigns parameters to samples, not features



Quadratic kernels

- While working in higher dimensions is beneficial, it also increases our running time because of the dot product computation
- However, there is a neat trick we can use
- consider all quadratic terms for $x^1, x^2 \ldots x^m$

$$\max_\alpha \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x_i})\Phi(\mathbf{x_j})$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

m is the number of features in each vector

The √2 term will become clear in the next slide

$$\Phi(x) = \begin{matrix} 1 \\ \sqrt{2}x^1 \\ \vdots \\ \sqrt{2}x^m \\ \\ (x^1)^2 \\ \vdots \\ (x^m)^2 \\ \\ \sqrt{2}x^1x^2 \\ \vdots \\ \sqrt{2}x^{m-1}x^m \end{matrix}$$

m+1 linear terms

m quadratic terms

m(m-1)/2 pairwise terms

Dot product for quadratic kernels

How many operations do we need for the dot product?

$$\Phi(x)\Phi(z) = \begin{pmatrix} 1 \\ \sqrt{2}x^1 \\ \vdots \\ \sqrt{2}x^1 \\ \\ (x^1)^2 \\ \vdots \\ (x^m)^2 \\ \\ \sqrt{2}x^1x^2 \\ \vdots \\ \sqrt{2}x^{m-1}x^m \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}z^1 \\ \vdots \\ \sqrt{2}z^2 \\ \\ (z^1)^2 \\ \vdots \\ (x^m)^2 \\ \\ \sqrt{2}z^1z^2 \\ \vdots \\ \sqrt{2}z^{m-1}z^m \end{pmatrix} = \sum_i 2x^i z^i + \sum_i (x^i)^2 (z^i)^2 + \sum_i \sum_{j=i+1} 2x^i x^j z^i z^j + 1$$

m m m(m-1)/2 **=~ m²**

The kernel trick

How many operations do we need for the dot product?

$$= \sum_i 2x^i z^i + \sum_i (x^i)^2 (z^i)^2 + \sum_i \sum_{j=i+1} 2x^i x^j z^i z^j + 1$$

m m m(m-1)/2 **=~ m^2**

However, we can obtain dramatic savings by noting that

$$\begin{aligned}(x.z+1)^2 &= (x.z)^2 + 2(x.z) + 1 \\ &= (\sum_i x^i z^i)^2 + \sum_i 2x^i z^i + 1 \\ &= \sum_i 2x^i z^i + \sum_i (x^i)^2 (z^i)^2 + \sum_i \sum_{j=i+1} 2x^i x^j z^i z^j + 1\end{aligned}$$

We only need m operations!

Note that to evaluate a new sample we are also using dot products so we save there as well

Where we are

Our dual target function:

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i x_j}$$

$$\sum_i \alpha_i \mathrm{y_i} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

mn^2 operations at each iteration

To evaluate a new sample x_j we need to compute:

$$\mathrm{w}^{\mathrm{T}} x_j + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y_i} \mathbf{x_i x_j} + b$$

mr operations where *r* are the number of support vectors (α_i>0)

Other kernels

- The kernel trick works for higher order polynomials as well.
- For example, a polynomial of degree 4 can be computed using $(x.z+1)^4$ and, for a polynomial of degree d $(x.z+1)^d$
- Beyond polynomials there are other very high dimensional basis functions that can be made practical by finding the right Kernel Function

-Radial-Basis-style Kernel Function:

$$K(x,z) = \exp\left(-\frac{(x-z)^2}{2\sigma^2}\right)$$

- Neural-net-style Kernel Function:

$$K(x,z) = \tanh(\kappa x.z - \delta)$$

Dual formulation for non linearly separable case

Dual target function:

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i} \mathbf{x_j}$$

$$\sum_{i} \alpha_i \mathrm{y_i} = 0$$

$$C > \alpha_i \geq 0 \qquad \forall i$$

The only difference is that the α_l's are now bounded

To evaluate a new sample x_j we need to compute:

$$\mathrm{w}^{\mathrm{T}} x_j + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y_i} \mathbf{x_i} \mathbf{x_j} + b$$

Why do SVMs work?

- If we are using huge features spaces (with kernels) how come we are not overfitting the data?

  - Number of parameters remains the same (and most are set to 0)

  - While we have a lot of input values, at the end we only care about the support vectors and these are usually a small group of samples

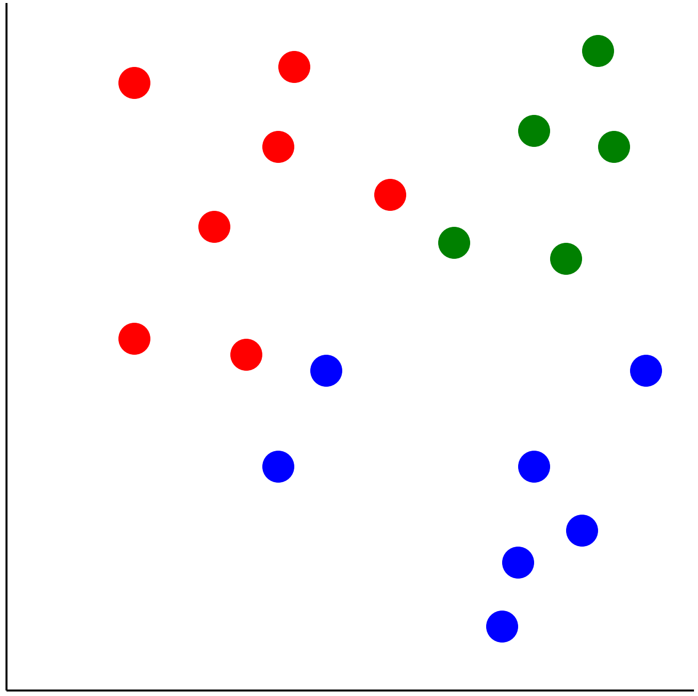
  - The minimization (or the maximizing of the margin) function acts as a sort of regularization term leading to reduced overfitting

Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Multi-class classification with SVMs

What if we have data from more than two classes?



- Most common solution: One vs. all

- create a classifier for each class against all other data

- for a new point use all classifiers and compare the margin for all selected classes

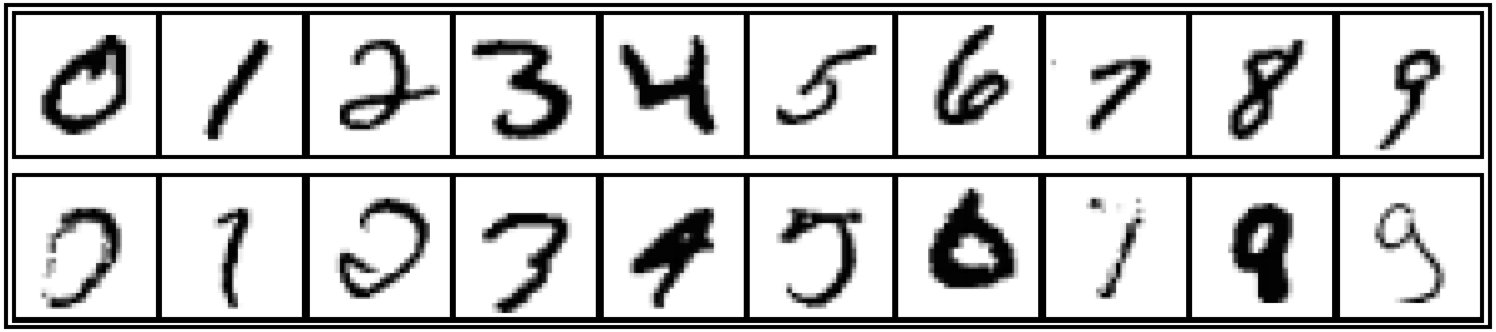
Note that this is not necessarily valid since this is not what we trained the SVM for, but often works well in practice

Applications of SVMs

- Bioinformatics
- Machine Vision
- Text Categorization
- Ranking (e.g., Google searches)
- Handwritten Character Recognition
- Time series analysis

→Lots of very successful applications!!!

Handwritten digit recognition



3-nearest-neighbor = 2.4% error

400–300–10 unit MLP = 1.6% error

LeNet: 768–192–30–10 unit MLP = 0.9% error

Current best (kernel machines, vision algorithms) ≈ 0.6% error

Important points

- Difference between regression classifiers and SVMs’
- Maximum margin principle
- Target function for SVMs
- Linearly separable and non separable cases
- Dual formulation of SVMs
- Kernel trick and computational complexity