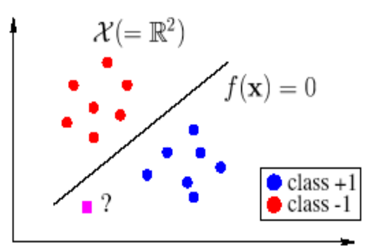
Introduction to Kernel Methods

Classifying data

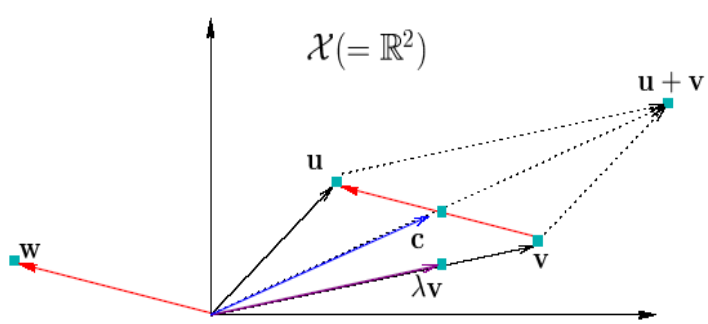
- Important notions in *learning to classify*
  - limited number of *training* data (patients, sequences, molecules, etc.)
  - learning algorithm (how to build the classifier?)
  - generalization: the classifier should correctly classify *test* data
- Quick formalization
  - $\mathcal{X}$ (e.g. $\mathbb{R}^d, d > 0$) is the space of data, called *input space*
  - $\mathcal{Y}$ (e.g. toxic/not toxic, or $\{-1, +1\}$) is the target space
  - $f : \mathcal{X} \to \mathcal{Y}$ is the classifier



Notion of Similarity

- Given a test data $x \in X$ we choose y such that (x, y) is in some sense similar to the training examples (e.g. *k-NN*).
- Thus we need a notion of similarity in X and in $\{\pm 1\}$
- **The choice of the similarity measure for the inputs is a deep question that lie at the core of machine learning.**
- *A simple type of similarity measure is the dot product (inner product or scalar product).*

Vectors and dot product

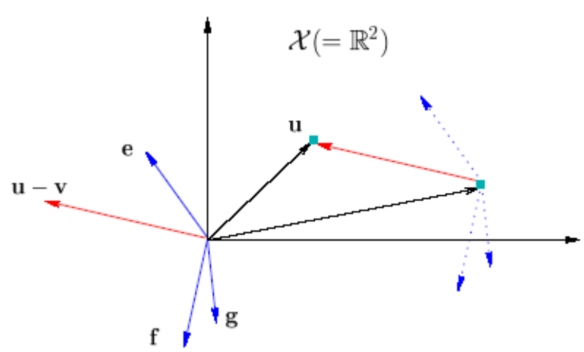


- $\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{c}$ are vectors
- $\mathbf{w} = \mathbf{u} - \mathbf{v}$ (red arrows)
- $\mathbf{c} = \frac{1}{2}(\mathbf{u}+\mathbf{v})$
- Here: $0 < \lambda < 1$

Vectors and dot product

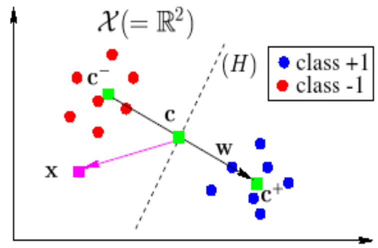
- Inner product $\langle \cdot, \cdot \rangle : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$:
  - symmetric: $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$
  - bilinear: $\langle \lambda \boldsymbol{u}_1 + \gamma \boldsymbol{u}_2, \boldsymbol{v} \rangle = \lambda \langle \mathbf{u}_1, \mathbf{v} \rangle + \gamma \langle \mathbf{u}_2, \mathbf{v} \rangle$
  - positive: $\langle \mathbf{u}, \mathbf{u} \rangle \geq 0$
  - definite: $\langle \mathbf{u}, \mathbf{u} \rangle = 0 \Rightarrow \mathbf{u} = 0$
- An inner product
  - provides $\mathcal{X}$ with a structure
  - can be viewed as a 'similarity'
  - defines a norm $\|\cdot\|$ on $\mathcal{X}$: $\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}$
- Example in $\mathbb{R}^2$
  - $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$: $\langle \mathbf{u}, \mathbf{v} \rangle = u_1 v_1 + u_2 v_2$

Vectors and dot product



- $\langle \mathbf{u} - \mathbf{v}, \mathbf{e} \rangle > 0$: $\mathbf{u} - \mathbf{v}$ and $\mathbf{e}$ point to the 'same direction'
- $\langle \mathbf{u} - \mathbf{v}, \mathbf{f} \rangle = 0$: $\mathbf{u} - \mathbf{v}$ and $\mathbf{f}$ are orthogonal
- $\langle \mathbf{u} - \mathbf{v}, \mathbf{g} \rangle < 0$: $\mathbf{u} - \mathbf{v}$ and $\mathbf{g}$ point to 'opposite directions'

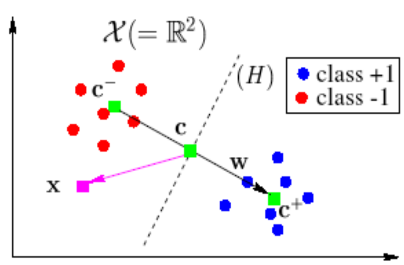
A simple linear classifier



- $\mathbf{c}^+ = \frac{1}{m^+} \sum_{\{i:y_i=+1\}} \mathbf{x}_i$
- $\mathbf{c}^- = \frac{1}{m^-} \sum_{\{i:y_i=-1\}} \mathbf{x}_i$
- $\mathbf{c} = \frac{1}{2}(c^+ + c^-)$
- $\mathbf{w} = c^+ - c^-$

- **Idea: assign a new point to the class whose mean is the closest.**
  - for $\mathbf{x} \in \mathcal{X}$, it is sufficient to take the sign of the inner product between $\mathbf{w}$ and $\mathbf{x} - \mathbf{c}$
  - if $h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} - \mathbf{c} \rangle$, we have the classifier $f(\mathbf{x}) =$ sign $(h(\mathbf{x}))$
  - the (dotted) hyperplane (H), of normal vector $\mathbf{w}$, is the decision surface

A simple linear classifier



- $\mathbf{c}^+ = \frac{1}{m^+} \sum_{\{i:y_i=+1\}} \mathbf{x}_i$
- $\mathbf{c}^- = \frac{1}{m^-} \sum_{\{i:y_i=-1\}} \mathbf{x}_i$
- $\mathbf{c} = \frac{1}{2}(c^+ + c^-)$
- $\mathbf{w} = c^+ - c^-$

- On evaluating $h(\mathbf{x})$

$$h(x) = \langle w, x - c \rangle = \langle w, x \rangle - \langle w, c \rangle = \langle c^+ - c^-, x \rangle - \langle c^+ - c^-, c \rangle$$

$$= \langle x, c^+ \rangle - \langle x, c^- \rangle - \langle c, c^+ \rangle + \langle c, c^- \rangle$$

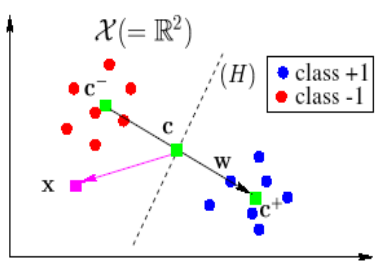
$$= \langle x, c^+ \rangle - \langle x, c^- \rangle + b \quad \text{where} \quad b = \langle c, c^- \rangle - \langle c, c^+ \rangle$$

A simple linear classifier

$$h(x) = \langle x, c^+ \rangle - \langle x, c^- \rangle + b$$
$$= \left\langle x, \frac{1}{m^+} \sum_{i:y_i=1} x_i \right\rangle - \left\langle x, \frac{1}{m^-} \sum_{i:y_i=-1} x_i \right\rangle + b$$
$$= \frac{1}{m^+} \sum_{i:y_i=1} \langle x, x_i \rangle - \frac{1}{m^-} \sum_{i:y_i=-1} \langle x, x_i \rangle + b$$
$$= \sum_{i=1}^{m} \alpha_i \langle x, x_i \rangle + b$$

where $\alpha_i = \frac{1}{m^+} \forall i : y_i = 1$ ***and*** $\alpha_i = \frac{1}{m^-} \forall i : y_i = -1$

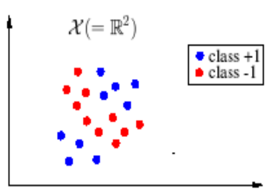
A simple linear classifier



- $\mathbf{c}^+ = \frac{1}{m^+} \sum_{\{i:y_i=+1\}} \mathbf{x}_i$
- $\mathbf{c}^- = \frac{1}{m^-} \sum_{\{i:y_i=-1\}} \mathbf{x}_i$
- $\mathbf{c} = \frac{1}{2}(c^+ + c^-)$
- $\mathbf{w} = c^+ - c^-$

- To summarize: $\boxed{h(\mathbf{x}) = \sum_{i=1,\dots,m} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b}$

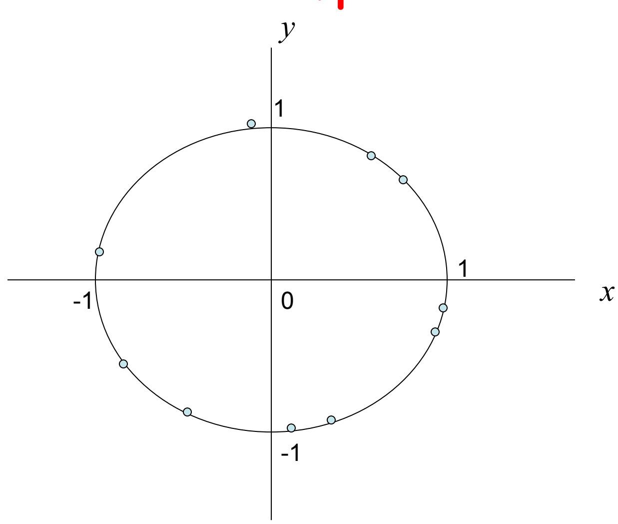
- Question: what if the dataset is not linearly separable, i.e. (H) fails to separate red and blue disks?



Non-linear Patterns in Data: an example

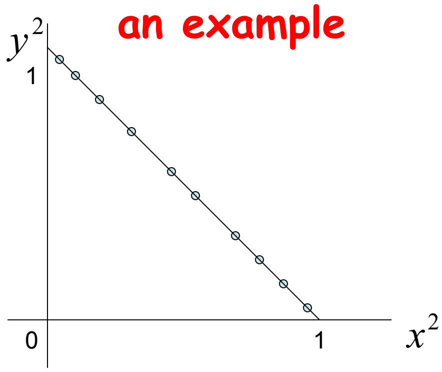
x	y	x^2	y^2	xy
0.8415	0.5403	0.7081	0.2919	0.4546
0.9093	-0.4161	0.8268	0.1732	-0.3784
0.1411	-0.99	0.0199	0.9801	-0.1397
-0.7568	-0.6536	0.5728	0.4272	0.4947
-0.9589	0.2837	0.9195	0.0805	-0.272
-0.2794	0.9602	0.0781	0.9219	-0.2683
0.657	0.7539	0.4316	0.5684	0.4953
0.9894	-0.1455	0.9788	0.0212	-0.144
0.4121	-0.9111	0.1698	0.8302	-0.3755
-0.544	-0.8391	0.296	0.704	0.4565

Non-linear Patterns in Data: an example



Data in the (x,y) plane

Non-linear Patterns in Data: an example



By changing the coordinate system the relation has become *linear*

Non-linear Patterns in Data: an example

- Using the initial coordinates, the pattern was expressed as a *quadratic* form:

$$f(\boldsymbol{x}) = x^2 + y^2 - 1 = 0 \qquad \forall\, \boldsymbol{x}$$

- In the coordinate system using monomials, it appeared as a *linear* function.

- *The possibility of transforming the representation of a pattern by changing the coordinate system in which the data are described is a recurrent theme in kernel methods.*

The Kernel trick

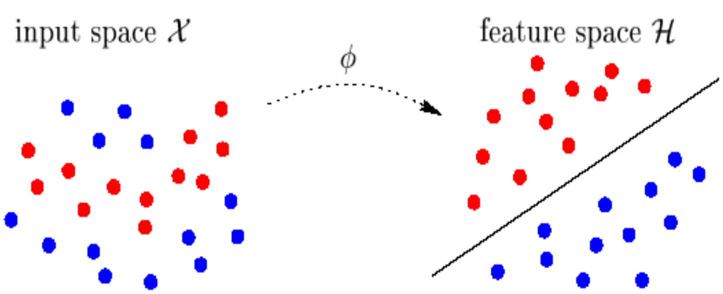
- Context: nonlinearly separable dataset $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$
- Idea to learn a nonlinear classifier
  - choose a (nonlinear) mapping ϕ

$$\begin{array}{rccc} \phi: & \mathcal{X} & \rightarrow & \mathcal{H} \\ & \mathbf{x} & \mapsto & \phi(\mathbf{x}) \end{array}$$

    where $\mathcal{H}$ is an inner product space (inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$), called *feature space*
  - find a linear classifier (i.e. a separating hyperplane) in $\mathcal{H}$ to classify $\{(\phi(\mathbf{x}_1), y_1), \dots, (\phi(\mathbf{x}_m), y_m)\}$

The Kernel trick

- Linearly classifying in *feature space*



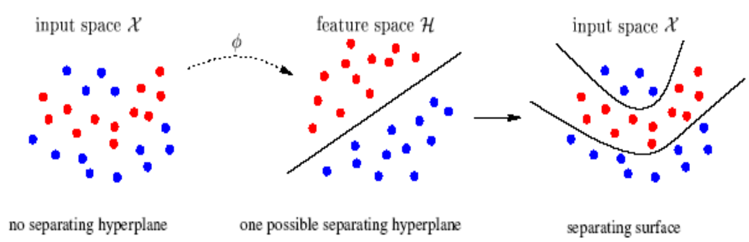
- Taking the previous linear algorithm and implementing it in $\mathcal{H}$:

$$h(\mathbf{x}) = \sum_{i=1,\dots,m} \alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle_{\mathcal{H}} + b$$

The Kernel trick

- The kernel trick can be applied if there is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that: $k(\mathbf{u}, \mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle_{\mathcal{H}}$
  If so, all occurrences of $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle_{\mathcal{H}}$ are replaced by $k(\mathbf{x}_i, \mathbf{x})$
- Keypoint: the 'focus' is sometimes only on k and not on ϕ
- Kernels must verify Mercer's property to be valid kernels
  - ensures that there exist a space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that $k(\mathbf{u}, \mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle_{\mathcal{H}}$
  - however non valid kernels have been used with success
  - and, research is in progress on using non semi-definite kernels
- k might be viewed as a similarity measure

The Kernel trick



- *Kernel trick* recipe
  - consider a nonlinear classification problem on $\mathcal{X} \times \mathcal{Y}$
  - choose a linear classification algorithm (expr. in terms $\langle \cdot, \cdot \rangle$)
  - replace all occurrences of $\langle \cdot, \cdot \rangle$ by a kernel $k(\cdot, \cdot)$
- Obtained classifier: $f(\mathbf{x}) = \text{sign}\left(\sum_{i=1,\ldots,m} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \right)$

Common Kernels

- Gaussian kernel
  - $k(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma^2}\right), \quad \sigma^2 > 0$
  - the corresponding $\mathcal{H}$ is of infinite dimension
- Polynomial kernel
  - $k(\mathbf{u},\mathbf{v}) = (\langle \mathbf{u},\mathbf{v}\rangle + c)^d, \quad c \in \mathbb{R}, d \in \mathbb{N}$
  - a corresponding analytic ϕ may be constructed (see below)

Common Kernels

- Let $k = \langle \mathbf{u},\mathbf{v}\rangle^2_{\mathbb{R}^2}$ (polynomial kernel with $c = 0$ and $d = 2$) defined on $\mathbb{R}^2 \times \mathbb{R}^2$
- Consider the mapping:

$$\begin{array}{cccc} \phi: & \mathbb{R}^2 & \to & \mathbb{R}^3 \\ & \mathbf{x} = [x_1, x_2]^\top & \mapsto & \phi(\mathbf{x}) = \left[x_1^2, \sqrt{2}x_1x_2, x_2^2\right]^\top \end{array}$$

- We have, for $\mathbf{u},\mathbf{v} \in \mathbb{R}^2$:

$$\begin{aligned} \langle \phi(\mathbf{u}), \phi(\mathbf{v})\rangle_{\mathbb{R}^3} &= \langle [u_1^2, \sqrt{2}u_1u_2, u_2^2]^\top, [v_1^2, \sqrt{2}v_1v_2, v_2^2]^\top\rangle \\ &= (u_1v_1 + u_2v_2)^2 \\ &= \langle \mathbf{u},\mathbf{v}\rangle^2_{\mathbb{R}^2} \\ &= k(\mathbf{u},\mathbf{v}) \end{aligned}$$

Detecting Patterns via Kernel Methods

- The focus is on the use of patterns that are determined by *linear functions* in a suitably chosen feature space;
- Transforming the original dataset involves then selecting a feature space for the linear functions.

Advantages of linear functions:

- We can specify the feature space in an indirect but very natural way through the so-called *kernel function*;
- It enables us to use feature spaces whose dimensionality is more than polynomial in the relevant parameters, even though the computational cost remains polynomial.

Detecting Patterns via Kernel Methods

Pattern analysis is then a two-stage process:

- First, we must recode the data so that the patterns become representable with linear functions.
- Second, we can apply one of the standard linear pattern analysis algorithms to the transformed data.
- The resulting class of pattern analysis algorithms will be referred to as *kernel methods*.

Key aspects of Kernel Methods

- Data are embedded into a vector space called the feature space;
- Linear relations are sought among the images of the data in the feature space;
- The algorithms are implemented in such a way that the coordinates of the embedded points are not needed; only their pair-wise inner products are;
- The pair-wise inner products can be computed efficiently directly from the original data using a kernel function.

Useful link

- Kernel Machines: http://www.kernel-machines.org/

References

- J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Pattern analysis (Chapter 1).
- B. Scholkopf and A. Smola, *Learning with Kernels*. A Tutorial Introduction (Chapter 1). MIT University Press.