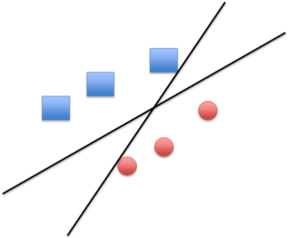
Kernel methods

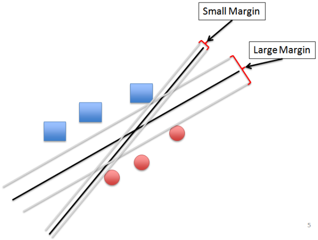
Vincent Barra

LIMOS, UMR CNRS 6158, Blaise Pascal University, Clermont-Ferrand, FRANCE

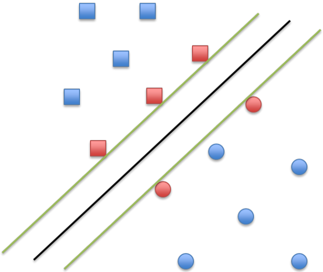
December 19, 2015



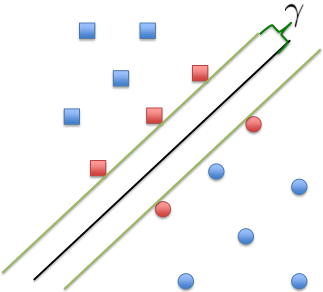
- Perceptron (and other linear classifiers) can lead to many equally valid choices for the decision boundary
- Are these really equally valid ?
- How can we pick which is best?



- Perceptron (and other linear classifiers) can lead to many equally valid choices for the decision boundary
- Are these really equally valid ?
- How can we pick which is best?
  $\rightarrow$ Maximize the size of the margin



- Support Vectors are those input points (vectors) closest to the decision boundary
- decision problem: $w^T x + b = 0$



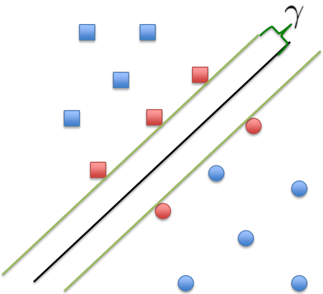
Notations

- x_i: data ; $y_i \in \{-1, 1\}$: labels
- decision hyperplane: $w^T x + b = 0$
- decision function : $D(x_i) = Sign(w^T x + b)$
- Margin hyperplanes: $w^T x + b = \pm\gamma$
- Scale invariance: $cw^T x + cb = 0$.

Scaling

This scaling does not change the decision hyperplane, or the support vector hyperplanes. But we will eliminate a variable from the optimization

$\Rightarrow$ Margin hyperplanes: $w^T x + b = \pm 1$



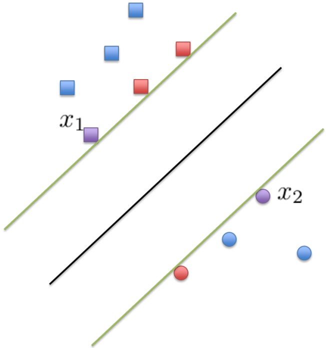
Notations

- x_i: data ; $y_i \in \{-1, 1\}$: labels
- decision hyperplane: $w^T x + b = 0$
- decision function :
  $D(x_i) = Sign(w^T x + b)$
- Margin hyperplanes: $w^T x + b = \pm\gamma$
- Scale invariance: $cw^T x + cb = 0$.

Scaling

This scaling does not change the decision hyperplane, or the support vector hyperplanes. But we will eliminate a variable from the optimization
$\Rightarrow$ Margin hyperplanes: $w^T x + b = \pm 1$

What are we optimizing ?



Size of the margin

represented in terms of w.

1. identification of a decision boundary
2. and simultaneously maximization of the margin

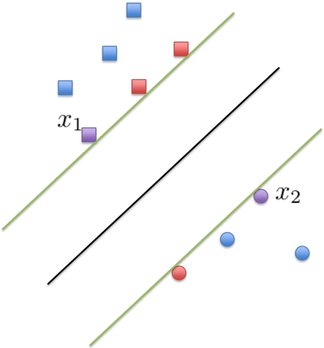
Relation margin $\leftrightarrow w$

There must at least one point that lies on each support hyperplanes. Thus

$$\rightarrow \; w^T x_1 + b = 1$$
$$\rightarrow \; w^T x_2 + b = -1$$

$$\Rightarrow w^T(x_1 - x_2) = 2$$

What are we optimizing ?



Size of the margin

represented in terms of w.

1. identification of a decision boundary
2. and simultaneously maximization of the margin

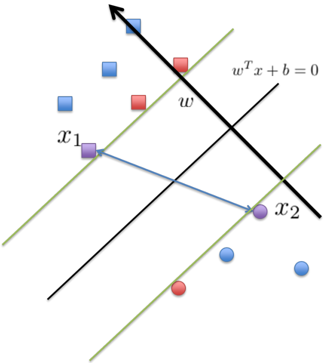
Relation margin $\leftrightarrow$ w

There must at least one point that lies on each support hyperplanes. Thus

$$\rightarrow \; w^T x_1 + b = 1$$
$$\rightarrow \; w^T x_2 + b = -1$$

$$\Rightarrow w^T(x_1 - x_2) = 2$$

WHAT ARE WE OPTIMIZING ?



$w^T(x_1 - x_2) = 2$

1. w is orthogonal to the decision hyperplane
2. margin: projection of $x_1 - x_2$ onto w,

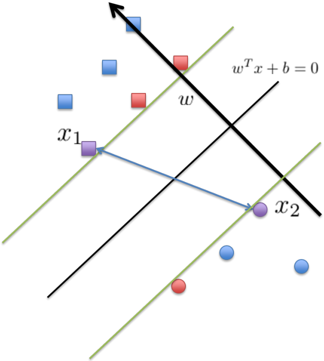
PROJECTION

$$w^T(x_1 - x_2) = 2$$

Projection: $\frac{w^T(x_1 - x_2)}{\|w\|} w$

Size of the margin: $\frac{2}{\|w\|}$

What are we optimizing ?



$w^T(x_1 - x_2) = 2$

1. w is orthogonal to the decision hyperplane
2. margin: projection of $x_1 - x_2$ onto w,

Projection

$$w^T(x_1 - x_2) = 2$$

Projection: $\frac{w^T(x_1 - x_2)}{||w||} w$

Size of the margin: $\frac{2}{||w||}$

MAXIMIZATION

$$Max \frac{2}{||w||}$$

subject to

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

MINIMIZATION

$$Min ||w||$$

subject to

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

LAGRANGIAN RELAXATION

$$L(w, b) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \left[y_i(w^T x_i + b) - 1 \right]$$

Maximization

$$Max \frac{2}{\|w\|}$$

subject to

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

Minimization

$$Min \|w\|$$

subject to

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

Lagrangian relaxation

$$L(w, b) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \left[y_i(w^T x_i + b) - 1 \right]$$

Maximization

$$Max \frac{2}{\|w\|}$$

subject to

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

Minimization

$$Min \|w\|$$

subject to

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

Lagrangian relaxation

$$L(w, b) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \left[y_i(w^T x_i + b) - 1 \right]$$

Primal problem

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$$

Dual problem

Now have to find α_i: substitute back to the Loss function

$$L(w,b) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \left[y_i (w^T x_i + b) - 1 \right]$$

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

where $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$

Primal problem

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$$

Dual problem

Now have to find α_i: substitute back to the Loss function

$$L(w, b) = \tfrac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \left[y_i (w^T x_i + b) - 1 \right]$$

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

where $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$

Primal problem

Optimize this quadratic program to identify the lagrange multipliers and thus the weights

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

where $\alpha_i \geq 0$

Support Vector Expansion

$$\begin{aligned} D(x) &= Sign(w^T x + b) \\ &= Sign\left(\left[\sum_{i=1}^{N} \alpha_i y_i x_i\right]^T x + b\right) \\ &= Sign\left(\left[\sum_{i=1}^{N} \alpha_i y_i x_i^T x\right] + b\right) \end{aligned}$$

- When α_i is non-zero then x_i is a support vector
- When α_i is zero x_i is not a support vector

Remark: $w = \sum^{N} \alpha_i y_i x_i$ Independent of the dimension of x_i

PRIMAL PROBLEM

Optimize this quadratic program to identify the lagrange multipliers and thus the weights

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

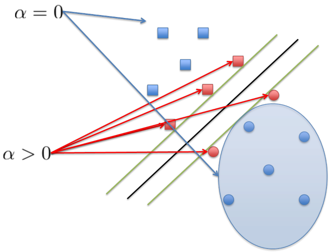
where $\alpha_i \geq 0$

SUPPORT VECTOR EXPANSION

$$\begin{aligned} D(x) &= Sign(w^T x + b) \\ &= Sign\left(\left[\sum_{i=1}^{N} \alpha_i y_i x_i \right]^T x + b \right) \\ &= Sign\left(\left[\sum_{i=1}^{N} \alpha_i y_i x_i^T x \right] + b \right) \end{aligned}$$

- When α_i is non-zero then x_i is a support vector
- When α_i is zero x_i is not a support vector

Remark: $w = \sum^{N} \alpha_i y_i x_i$ Independent of the dimension of x_i

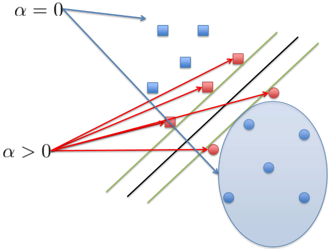


At the optimal solution

$$\alpha_i(1 - y_i(w^T x_i + b)) = 0$$

If $\alpha_i \neq 0 : y_i(w^T x_i + b) = 1$

$\Rightarrow$ Only points on the decision boundary contribute to the solution.



At the optimal solution

$$\alpha_i(1 - y_i(w^T x_i + b)) = 0$$

If $\alpha_i \neq 0 : y_i(w^T x_i + b) = 1$

$\Rightarrow$ Only points on the decision boundary contribute to the solution.

- α_i large $\Rightarrow$ the associated data point is quite important.
- It's either an outlier, or incredibly important

But this only gives us the best solution for linearly separable data sets

BOUNDS

Theoretical bounds on testing error:

- → The upper bound doesn't depend on the dimensionality of the space
- → The lower bound is maximized by maximizing the margin associated with the decision boundary

PROPERTIES OF SVM

- → Good generalization capability
- → Decision boundary is based on the data in the form of the support vectors → easy to interpret
- → Principled bounds on testing error from Learning Theory (VC dimension)

PROS AND CONS: SVM VS. MLP

- → SVMs have many fewer parameters
- → SVMs can be applied to non vectorial (or high dimensional) data
- → SVM: Convex optimization task / MLP local minima (likelihood non-convex)
- → SVM: Not especially fast.
  - Training: $O(n^3)$ (quadratic programming efficiency)
  - Evaluation: $O(n)$: Need to evaluate against each support vector

BOUNDS

Theoretical bounds on testing error:

- → The upper bound doesn't depend on the dimensionality of the space
- → The lower bound is maximized by maximizing the margin associated with the decision boundary

PROPERTIES OF SVM

- → Good generalization capability
- → Decision boundary is based on the data in the form of the support vectors → easy to interpret
- → Principled bounds on testing error from Learning Theory (VC dimension)

PROS AND CONS: SVM VS. MLP

- → SVMs have many fewer parameters
- → SVMs can be applied to non vectorial (or high dimensional) data
- → SVM: Convex optimization task / MLP local minima (likelihood non-convex)
- → SVM: Not especially fast.
  - Training: $O(n^3)$ (quadratic programming efficiency)
  - Evaluation: $O(n)$: Need to evaluate against each support vector

Bounds

Theoretical bounds on testing error:

- → The upper bound doesn't depend on the dimensionality of the space
- → The lower bound is maximized by maximizing the margin associated with the decision boundary

Properties of SVM

- → Good generalization capability
- → Decision boundary is based on the data in the form of the support vectors → easy to interpret
- → Principled bounds on testing error from Learning Theory (VC dimension)

Pros and cons: SVM vs. MLP

- → SVMs have many fewer parameters
- → SVMs can be applied to non vectorial (or high dimensional) data
- → SVM: Convex optimization task / MLP local minima (likelihood non-convex)
- → SVM: Not especially fast.
  - Training: $O(n^3)$ (quadratic programming efficiency)
  - Evaluation: $O(n)$: Need to evaluate against each support vector

OUTLIERS

→ There can be outliers on the other side of the decision boundary, or leading to a small margin.
→ Solution: Introduce a penalty term to the constraint function

NEW FUNCTION

$$Min||w|| + C\sum_{i=1}^{N}\xi_i$$

s.c.

$$\begin{aligned} y_i(w^T x_i + b) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}$$

$$L(w,b) = \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i(w^T x_i + b) + \xi_i - 1\right]$$

OUTLIERS

- → There can be outliers on the other side of the decision boundary, or leading to a small margin.
- → Solution: Introduce a penalty term to the constraint function

NEW FUNCTION

$$Min\|w\| + C\sum_{i=1}^{N}\xi_i$$

s.c.

$$\begin{aligned} y_i(w^T x_i + b) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}$$

$$L(w,b) = \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i(w^T x_i + b) + \xi_i - 1\right]$$

New function

$$Min\|w\| + C\sum_{i=1}^{N}\xi_i$$

s.c.

$$\begin{aligned} y_i(w^Tx_i + b) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}$$

$$L(w,b) = \frac{1}{2}w^Tw + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i(w^Tx_i + b) + \xi_i - 1\right]$$

Still quadratic programming

$$W(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}y_iy_j\alpha_i\alpha_jx_i^Tx_j$$

where $0 \leq \alpha_i \leq C$

$$\sum_{i=1}^{N}\alpha_iy_i = 0$$

NEW FUNCTION

$$Min\|w\| + C\sum_{i=1}^{N}\xi_i$$

s.c.

$$\begin{aligned} y_i(w^T x_i + b) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}$$

$$L(w,b) = \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i(w^T x_i + b) + \xi_i - 1\right]$$

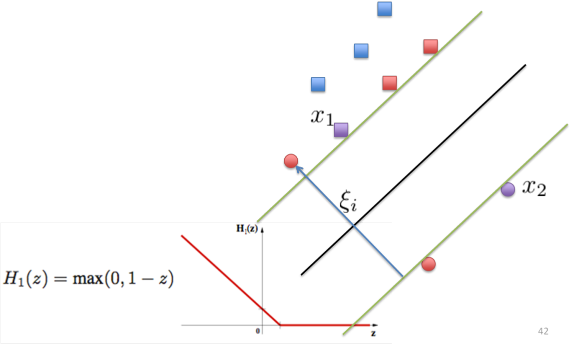
STILL QUADRATIC PROGRAMMING

$$W(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j$$

where $0 \leq \alpha_i \leq C$

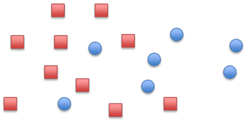
$$\sum_{i=1}^{N}\alpha_i y_i = 0$$

Soft margin Example



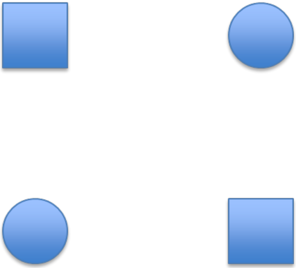
H_1: hinge loss

SVM: NON LINEARLY SEPARABLE CASE



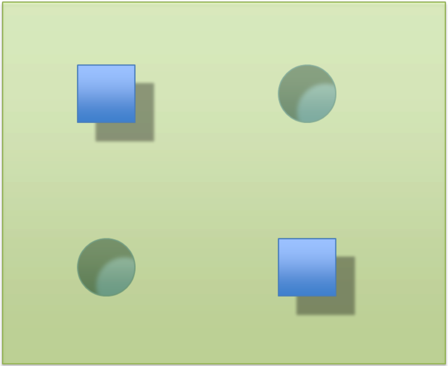
- → So far, support vector machines can only handle linearly separable data
- → But most data isn't.
- → We already see how to deal with this problem: soft margin
- → Now: another solution...

SVM: NON LINEARLY SEPARABLE CASE



→ Points that are not linearly separable in 2 dimension ..

SVM: NON LINEARLY SEPARABLE CASE



→ Points that are not linearly separable in 2 dimension, might be linearly separable in 3.

Recall...

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \; w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

And then...

- → The decision process doesn't depend on the dimensionality of the data.
- → We can map to a higher dimensionality of the data space.
- → data points only appear within a dot product.
- → The error is based on the dot product of data points, not the data points themselves.

And so...

How to add dimensionality to the data in order to make it linearly separable ?

- Extreme case: construct a dimension for each data point ⇒ overfitting
- Mapping: $x_i^T x_j \leftrightarrow \phi(x_i)^T \phi(x_j)$

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$$

Recall...

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \; w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

And then...

→ The decision process doesn't depend on the dimensionality of the data.
→ We can map to a higher dimensionality of the data space.
→ data points only appear within a dot product.
→ The error is based on the dot product of data points, not the data points themselves.

And so...

How to add dimensionality to the data in order to make it linearly separable ?

- Extreme case: construct a dimension for each data point ⇒ overfitting
- Mapping: $x_i^T x_j \leftrightarrow \phi(x_i)^T \phi(x_j)$

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$$

Recall...

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \; w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

And then...

→ The decision process doesn't depend on the dimensionality of the data.

→ We can map to a higher dimensionality of the data space.

→ data points only appear within a dot product.

→ The error is based on the dot product of data points, not the data points themselves.

And so...

How to add dimensionality to the data in order to make it linearly separable ?

- Extreme case: construct a dimension for each data point ⇒ overfitting
- Mapping: $x_i^T x_j \leftrightarrow \phi(x_i)^T \phi(x_j)$

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$$

Untractable example

$$\phi(x_0, x_1) = (x_0^2, x_0 x_1, x_1 x_0, x_1^2)$$

applied to a 20x30 image of 600 pixels $\approx$ 180000 dimensions !

Would be computationally infeasible to work in this space

Dual problem

- α_i: dual variables
- Since any component orthogonal to the space spanned by the training data has no effect, general result that weight vectors have dual representation: the representer theorem.
- can reformulate algorithms to learn dual variables rather than weight vector directly

UNTRACTABLE EXAMPLE

$$\phi(x_0, x_1) = (x_0^2, x_0x_1, x_1x_0, x_1^2)$$

applied to a 20x30 image of 600 pixels $\approx$ 180000 dimensions !

Would be computationally infeasible to work in this space

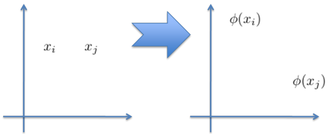
DUAL PROBLEM

- α_i: dual variables
- Since any component orthogonal to the space spanned by the training data has no effect, general result that weight vectors have dual representation: the representer theorem.
- can reformulate algorithms to learn dual variables rather than weight vector directly

KERNELS

1. We can represent this dot product as a Kernel (Kernel Function, Kernel Matrix)
2. Finite (if large) dimensionality of $K(x_i, x_j)$ unrelated to dimensionality of x

REMEMBER THE DUAL



Kernels are a mapping

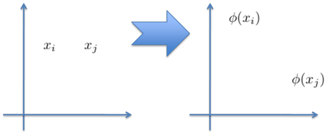
$$x_i^T x_j \leftrightarrow \phi(x_i)^T \phi(x_j)$$

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

Kernels

1. We can represent this dot product as a Kernel (Kernel Function, Kernel Matrix)
2. Finite (if large) dimensionality of $K(x_i, x_j)$ unrelated to dimensionality of x

Remember the dual



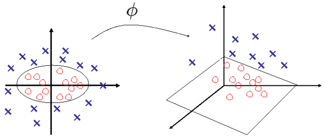
Kernels are a mapping

$$x_i^T x_j \leftrightarrow \phi(x_i)^T \phi(x_j)$$

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

Gram Matrix: $K_{ij} = K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

FIRST EXAMPLE



Consider the following Kernel:

$$\begin{aligned} K(x,z) &= (x^T z)^2 \\ &= (x_0 z_0 + x_1 z_1)^2 \\ &= x_0^2 z_0^2 + 2 x_0 z_0 x_1 z_1 + x_1^2 z_1^2 \\ &= (x_0^2, \sqrt{2} x_0 x_1, x_1^2)^T (z_0^2, \sqrt{2} z_0 z_1, z_1^2) \\ &= \phi(x)^T \phi(z) \end{aligned}$$

with

$$\phi(y) = (y_0^2, \sqrt{2} y_0 y_1, y_1^2)$$

SECOND EXAMPLE

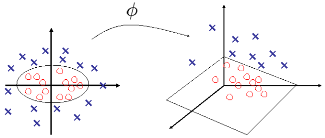
$\phi : x \in X \rightarrow \phi(x) \in \mathcal{F}$

$(x, y) \mapsto (x_0^2, x_0 x_1, x_1 x_0, x_1^2)$

Linear equation in $\mathcal{F}$ $a x_0^2 + b x_1^2 = c \rightarrow$ ellipse (non linear shape) in X

Gram Matrix: $K_{ij} = K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

FIRST EXAMPLE



Consider the following Kernel:

$$\begin{aligned} K(x,z) &= (x^T z)^2 \\ &= (x_0 z_0 + x_1 z_1)^2 \\ &= x_0^2 z_0^2 + 2x_0 z_0 x_1 z_1 + x_1^2 z_1^2 \\ &= (x_0^2, \sqrt{2} x_0 x_1, x_1^2)^T (z_0^2, \sqrt{2} z_0 z_1, z_1^2) \\ &= \phi(x)^T \phi(z) \end{aligned}$$

with

$$\phi(y) = (y_0^2, \sqrt{2} y_0 y_1, y_1^2)$$

SECOND EXAMPLE

$\phi : x \in X \to \phi(x) \in \mathcal{F}$
$(x, y) \mapsto (x_0^2, x_0 x_1, x_1 x_0, x_1^2)$
Linear equation in $\mathcal{F}$ $ax_0^2 + bx_1^2 = c \to$ ellipse (non linear shape) in X

The capacity is proportional to the dimension

Theorem

Given $m+1$ examples in general position in a m-dimensional space, every possible classification can be generated with a thresholded linear function

Extension: Cover's theorem

- Capacity may easily become too large and lead to over-fitting: being able to realise every classifier means unlikely to generalise well
- Computational costs involved in dealing with large vectors

Kernels

- In general: don't need to know the form of ϕ.
- Just specifying the kernel function is sufficient.
- A good kernel: Computing K_{ij} is cheaper than $\phi(x_i)$

Valid Kernels

- Symmetric
- Must be decomposable into ϕ functions
- Harder to show.
  - Gram matrix is positive semi-definite
  - Positive entries are definitely positive semi-definite.
  - Negative entries may still be positive semi-definite

$$x^T K x \geq 0$$

Examples

K, K' Kernel $\Rightarrow \quad cK, K + K', K.K', exp(K)...$
Examples: Polynomial kernels, RBF, String kernels, graph kernels
Note: a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

KERNELS

- In general: don't need to know the form of ϕ.
- Just specifying the kernel function is sufficient.
- A good kernel: Computing K_{ij} is cheaper than $\phi(x_i)$

VALID KERNELS

- Symmetric
- Must be decomposable into ϕ functions
- Harder to show.
  - Gram matrix is positive semi-definite
  - Positive entries are definitely positive semi-definite.
  - Negative entries may still be positive semi-definite

$$x^T Kx \geq 0$$

EXAMPLES

K, K' Kernel $\Rightarrow \quad cK, K+K', K.K', exp(K)...$
Examples: Polynomial kernels, RBF, String kernels, graph kernels
Note: a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

KERNELS

- In general: don't need to know the form of ϕ.
- Just specifying the kernel function is sufficient.
- A good kernel: Computing K_{ij} is cheaper than $\phi(x_i)$

VALID KERNELS

- Symmetric
- Must be decomposable into ϕ functions
- Harder to show.
  - Gram matrix is positive semi-definite
  - Positive entries are definitely positive semi-definite.
  - Negative entries may still be positive semi-definite

$$x^T K x \geq 0$$

EXAMPLES

K, K' Kernel $\Rightarrow \quad cK, K + K', K.K', exp(K)...$
Examples: Polynomial kernels, RBF, String kernels, graph kernels
Note: a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

$$
\begin{aligned}
W(\alpha) &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \\
&= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\end{aligned}
$$

- optimize the α_i and b w.r.t. K
- decision function $D(x) = sign \left[\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b \right]$

Polynomial Kernels

$$K(x,z) = (x^T z + c)^d \quad c \geq 0$$

- dot product $\rightarrow$ polynomial power of the original dot product.
- c large $\Rightarrow$ focus on linear terms
- c small $\Rightarrow$ focus on higher order terms
- Very fast to calculate

RBF

$$K(x,z) = e^{\frac{\|x-z\|^2}{2\sigma^2}}$$

- dot product $\rightarrow$ related to the distance in space between the two points.
- Placing a bump on each point

String kernels

Not a gaussian, but still a legitimate Kernel

- $K(s, s')$ = difference in length,count of different letters, minimum edit distance
- allow for infinite dimensional inputs
- don't need to manually encode the input

Polynomial Kernels

$$K(x,z) = (x^T z + c)^d \quad c \geq 0$$

- dot product $\rightarrow$ polynomial power of the original dot product.
- c large $\Rightarrow$ focus on linear terms
- c small $\Rightarrow$ focus on higher order terms
- Very fast to calculate

RBF

$$K(x,z) = e^{\frac{\|x-z\|^2}{2\sigma^2}}$$

- dot product $\rightarrow$ related to the distance in space between the two points.
- Placing a bump on each point

String kernels

Not a gaussian, but still a legitimate Kernel

- $K(s,s')$ = difference in length,count of different letters, minimum edit distance
- allow for infinite dimensional inputs
- don't need to manually encode the input

Polynomial Kernels

$$K(x,z) = (x^T z + c)^d \quad c \geq 0$$

- dot product $\rightarrow$ polynomial power of the original dot product.
- c large $\Rightarrow$ focus on linear terms
- c small $\Rightarrow$ focus on higher order terms
- Very fast to calculate

RBF

$$K(x,z) = e^{\frac{\|x-z\|^2}{2\sigma^2}}$$

- dot product $\rightarrow$ related to the distance in space between the two points.
- Placing a bump on each point

String kernels

Not a gaussian, but still a legitimate Kernel

- $K(s, s')$ = difference in length,count of different letters, minimum edit distance
- allow for infinite dimensional inputs
- don't need to manually encode the input

Graph Kernels

- Define the kernel function based on graph properties
- must be computable in poly-time (paths, spanning trees, cycles, bag of paths...)
- Possible incorporation of knowledge about the input without direct feature extraction

PAIRED CORPORA

- Can we use paired corpora to extract more information?
- Two views of same semantic object - hypothesise that both views contain all of the necessary information, eg document and translation to a second language:

$$\phi_a(d) \leftrightarrow d \leftrightarrow \phi_b(d)$$

EXAMPLE: CANADIAN PARLIAMENT CORPUS

LAND MINES (E_{12})
Ms. Beth Phinney (Hamilton Mountain, Lib.): Mr. Speaker, we are pleased that the Nobel peace prize has been given to those working to ban land mines worldwide. We hope this award will encourage the United States to join the over 100 countries planning to come to ...

LES MINES ANTIPERSONNEL (F_{12})
Mme Beth Phinney (Hamilton Mountain, Lib.): Monsieur le Prsident, nous nous rjouissons du fait que le prix Nobel ait t attribu ceux qui oeuvrent en faveur de l'interdiction des mines antipersonnel dans le monde entier. Nous esprons que cela incitera les Amricains se joindre aux reprsentants de plus de 100 pays qui ont l'intention de venir ...

PAIRED CORPORA

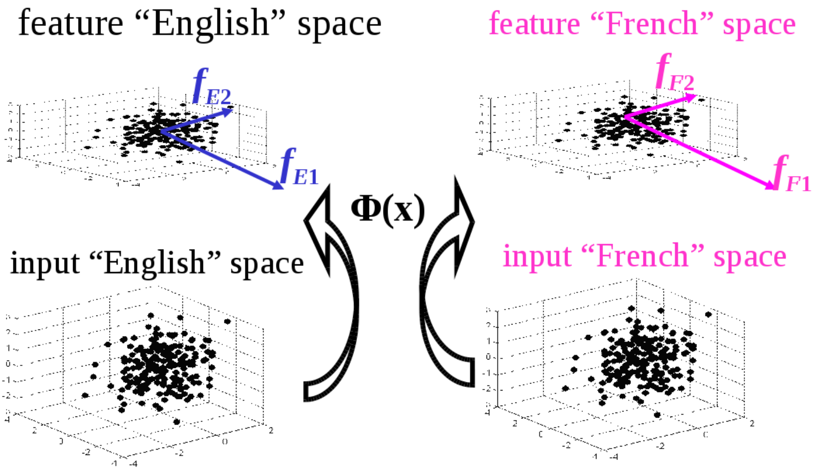
- Can we use paired corpora to extract more information?
- Two views of same semantic object - hypothesise that both views contain all of the necessary information, eg document and translation to a second language:

$$\phi_a(d) \leftrightarrow d \leftrightarrow \phi_b(d)$$

EXAMPLE: CANADIAN PARLIAMENT CORPUS

LAND MINES (E_{12})
Ms. Beth Phinney (Hamilton Mountain, Lib.): Mr. Speaker, we are pleased that the Nobel peace prize has been given to those working to ban land mines worldwide. We hope this award will encourage the United States to join the over 100 countries planning to come to ...

LES MINES ANTIPERSONNEL (F_{12})
Mme Beth Phinney (Hamilton Mountain, Lib.): Monsieur le Prsident, nous nous rjouissons du fait que le prix Nobel ait t attribu ceux qui oeuvrent en faveur de l'interdiction des mines antipersonnel dans le monde entier. Nous esprons que cela incitera les Amricains se joindre aux reprsentants de plus de 100 pays qui ont l'intention de venir ...


feature "English" space
f_{E2}
f_{E1}
feature "French" space
f_{F2}
f_{F1}
$\Phi(\mathbf{x})$
input "English" space
input "French" space

PAPRT

M. L. Littman, S. T. Dumais, and T. K. Landauer. Automatic cross-language information retrieval using latent semantic indexing. In G. Grefenstette, editor, Cross-language information retrieval. Kluwer, 1998.

FORMULATION

$$Max_{f_E, f_F} Corr\left(f_E^T \phi_E(E), f_F^T \phi_F(F)\right)$$

$$f_E = \sum_l \alpha_l \phi_E(E_l), f_F = \sum_l \beta_l \phi_F(F_l)$$

$$\begin{aligned} B\xi &= \rho D\xi \\ \begin{pmatrix} 0 & K_E K_F \\ K_F K_E & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} &= \rho \begin{pmatrix} K_E^2 & 0 \\ 0 & K_F^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \end{aligned}$$

REGULARIZATION

- using kernel functions may result in overfitting
- Theoretical analysis shows that provided the norms of the weight vectors are small the correlation will still hold for new data
- need to control flexibility of the projections f_E and f_F: add diagonal term κ to the matrix D

Paprt

M. L. Littman, S. T. Dumais, and T. K. Landauer. Automatic cross-language information retrieval using latent semantic indexing. In G. Grefenstette, editor, Cross-language information retrieval. Kluwer, 1998.

Formulation

$$Max_{f_E, f_F} Corr\left(f_E^T \phi_E(E), f_F^T \phi_F(F)\right)$$

$$f_E = \sum_l \alpha_l \phi_E(E_l), f_F = \sum_l \beta_l \phi_F(F_l)$$

$$\begin{aligned} B\xi &= \rho D\xi \\ \begin{pmatrix} 0 & K_E K_F \\ K_F K_E & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} &= \rho \begin{pmatrix} K_E^2 & 0 \\ 0 & K_F^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \end{aligned}$$

Regularization

- using kernel functions may result in overfitting
- Theoretical analysis shows that provided the norms of the weight vectors are small the correlation will still hold for new data
- need to control flexibility of the projections f_E and f_F: add diagonal term κ to the matrix D

Paprt

M. L. Littman, S. T. Dumais, and T. K. Landauer. Automatic cross-language information retrieval using latent semantic indexing. In G. Grefenstette, editor, Cross-language information retrieval. Kluwer, 1998.

Formulation

$$Max_{f_E,f_F} Corr\left(f_E^T \phi_E(E), f_F^T \phi_F(F)\right)$$

$$f_E = \sum_l \alpha_l \phi_E(E_l), f_F = \sum_l \beta_l \phi_F(F_l)$$

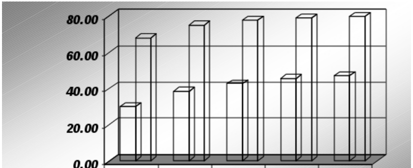
$$\begin{aligned} B\xi &= \rho D\xi \\ \begin{pmatrix} 0 & K_E K_F \\ K_F K_E & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} &= \rho \begin{pmatrix} K_E^2 & 0 \\ 0 & K_F^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \end{aligned}$$

Regularization

- using kernel functions may result in overfitting
- Theoretical analysis shows that provided the norms of the weight vectors are small the correlation will still hold for new data
- need to control flexibility of the projections f_E and f_F: add diagonal term κ to the matrix D

The goal was to retrieve the paired document.

1. LSI/KCCA trained on paired documents,
2. All test documents projected into the LSI/KCCA semantic space,
3. Each query was projected into the LSI/KCCA semantic space and documents were retrieved using nearest neighbour based on cosine distance to the query.



	100	200	300	400	500
CL-LSI	29.95	38.20	42.44	45.35	47.11
CL_KCCA	67.88	74.92	77.56	78.87	79.79

- Data
  - Combined image and associated text obtained from the web
  - Three categories: sport, aviation and paintball
  - 400 examples from each category (1200 overall)
  - Features extracted: HSV, Texture , Bag of words
- Tasks
  - Classification of web pages into the 3 categories
  - Text query $\rightarrow$ image retrieval

To conclude: kernel Trick

To conclude (Kernel trick) : a kernel can be applied where a dot product is used in an optimization:

- Kernel PCA
- Kernel perceptron
- unsupervised clustering (similarity $\approx$ distance $\leftrightarrow$ dot product)