Support Vector Machines

srihari@buffalo.edu

SVM Discussion Overview

1. Overview of SVMs
2. Margin Geometry
3. SVM Optimization
4. Overlapping Distributions
5. Relationship to Logistic Regression
6. Dealing with Multiple Classes
7. SVM for Regression
8. SVM and Computational Learning Theory
9. Relevance Vector Machines

Kernel Methods

- In the kernel method for regression
  - Given data set $\{\boldsymbol{x}_i, t_i\},\ i = 1..N$
    - Gram matrix $\boldsymbol{K}$ has elements

      $$K_{nm} = \phi(\boldsymbol{x}_n)^{\mathrm{T}} \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

  - Output y predicted for $\boldsymbol{x}$ is

    $y(\boldsymbol{x}) = k(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{K} + \lambda \mathrm{I}_{\mathrm{N}})^{-1}\boldsymbol{t}$, where $k(\boldsymbol{x})$ has elements $k_n(\boldsymbol{x}) = (\boldsymbol{x}_n, \boldsymbol{x})$

Primal optimization

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\boldsymbol{w}^T\phi(\mathrm{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\boldsymbol{w} = \Phi^T\boldsymbol{a} \qquad a_n = -\frac{1}{\lambda}\left\{\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - t_n\right\}$$

$\boldsymbol{w}$ appearing in $\boldsymbol{a}$ eliminated using kernel

$$J(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{a}^T\Phi\Phi^T\Phi\Phi^T\boldsymbol{a} - \boldsymbol{a}^T\Phi\Phi^T\mathrm{t} + \frac{1}{2}\mathrm{t}^T\mathrm{t} + \frac{\lambda}{2}\boldsymbol{a}^T\Phi\Phi^T\boldsymbol{a}$$

Dual optimization

$$J(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T KK\boldsymbol{a} - \boldsymbol{a}^T K\mathrm{t} + \frac{1}{2}\mathrm{t}^T\mathrm{t} + \frac{\lambda}{2}\boldsymbol{a}^T K\boldsymbol{a}$$

$$\boldsymbol{a} = (K + \lambda \mathrm{I}_{\mathrm{N}})^{-1}\mathrm{t}$$

- Many choices of kernel
  1. Gaussian: $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-||\boldsymbol{x} - \boldsymbol{x}'||^2 / 2\sigma^2)$
  2. Cosine similarity (BOW, TF-IDF): $$\boxed{k(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \frac{\boldsymbol{x}_i^T \boldsymbol{x}_{i'}}{||\boldsymbol{x}_i||_2 ||\boldsymbol{x}_{i'}||_2}}$$ $\boxed{\boldsymbol{x}_i = [x_{i1}, .., x_{i\mathrm{D}}]}$
  3. Mercer kernel: # substrings shared $$\boxed{k(x, x') = \sum_{s \in A^*} w_s \phi_s(x) \phi_s(x')}$$ $\phi_s(x)$ is no of times substring s appears in x

Sparse Kernel Methods

- Kernel methods have a limitation
  - that $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ must be evaluated for all possible pairs $(\boldsymbol{x}_n, \boldsymbol{x}_m)$ of training points
  - Which is computationally infeasible
- In sparse solutions, predictions for new inputs depend only on kernels evaluated at a subset of training data points

Alternative Names for SVM

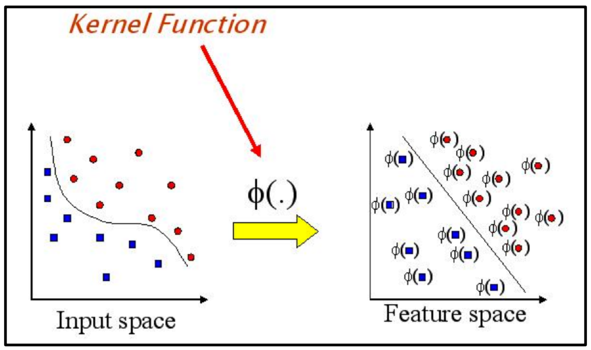
- Also called Sparse kernel machines
  - Prediction is based on a linear combination of a kernel evaluated at the training points
  - Sparse because not all pairs of training points need be used
- Also called Maximum margin classifiers

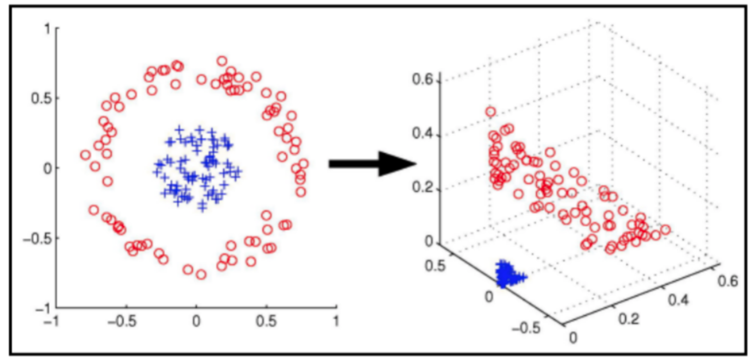
Importance of SVMs

- SVM is a discriminative method
- SVM brings together:
  1. Computational learning theory
  2. Previous methods in linear discriminants
  3. Optimization theory
- Widely used for ML problems:
  - Classification, regression, novelty detection

Ideas Used in SVM

- Linearly separable case considered
  - By using higher-dimensions than original feature space





- Kernel trick reduces computational overhead

$$k(\boldsymbol{y}_j,\boldsymbol{y}_k) = \boldsymbol{y}_j^{\,t}\cdot\boldsymbol{y}_k = \phi(\boldsymbol{x}_j)^t\cdot\phi(\boldsymbol{x}_k)$$

- Determination of model parameters is a convex optimization problem
  - Extensive use of Lagrange multipliers

2. Maximum Margin Classifiers

- Begin with 2-class linear classifier

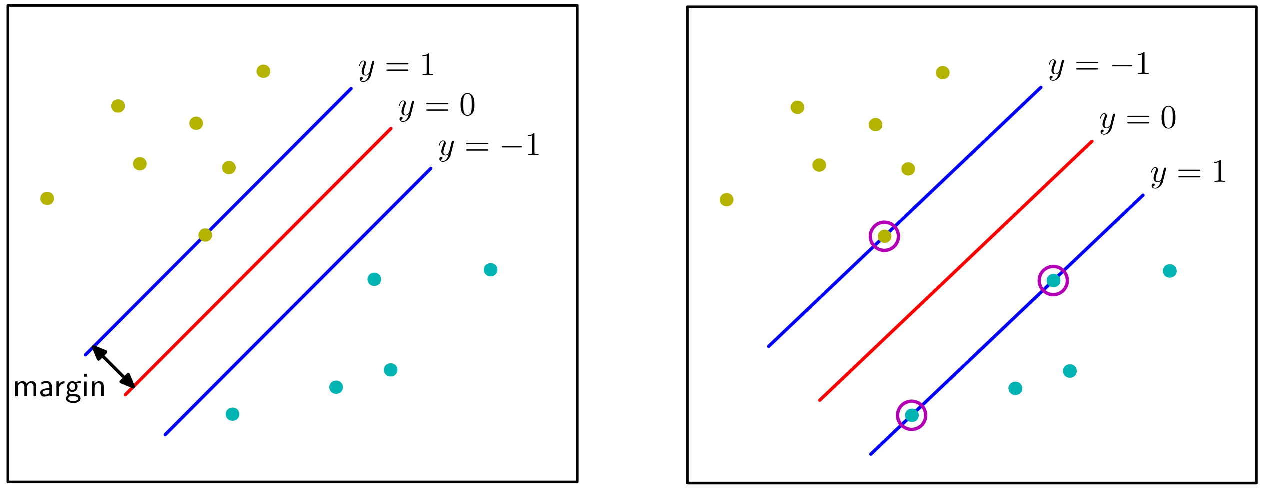
  $$y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b$$

  - where $\boldsymbol{\phi}(\boldsymbol{x})$ is a feature space transformation
  - We will introduce a dual representation which avoids working with feature space
    - Training set: $\boldsymbol{x}_1,..,\boldsymbol{x}_N$ with targets $t_1,..,t_N$, $t_n \, \varepsilon \, \{-1,1\}$
    - New $\boldsymbol{x}$ classified according to sign of $y(\boldsymbol{x})$
  - Assume training data is linearly separable in feature space,
    - i.e., there exists at least one choice of $\boldsymbol{w}$ and $\boldsymbol{b}$ such that $t_n y(\boldsymbol{x}_n) > 0$ for all points

Definition of margin

- Perceptron guarantees a solution in finite no. of steps, which depends on
  1. Initial values for $\boldsymbol{w}, b$
  2. Order in which data are presented
- If there are multiple solutions we need one with smallest generalization error
  - SVM approaches this with margin concept
- Defined as smallest distance between decision boundary and any of the samples

Support Vector Definition



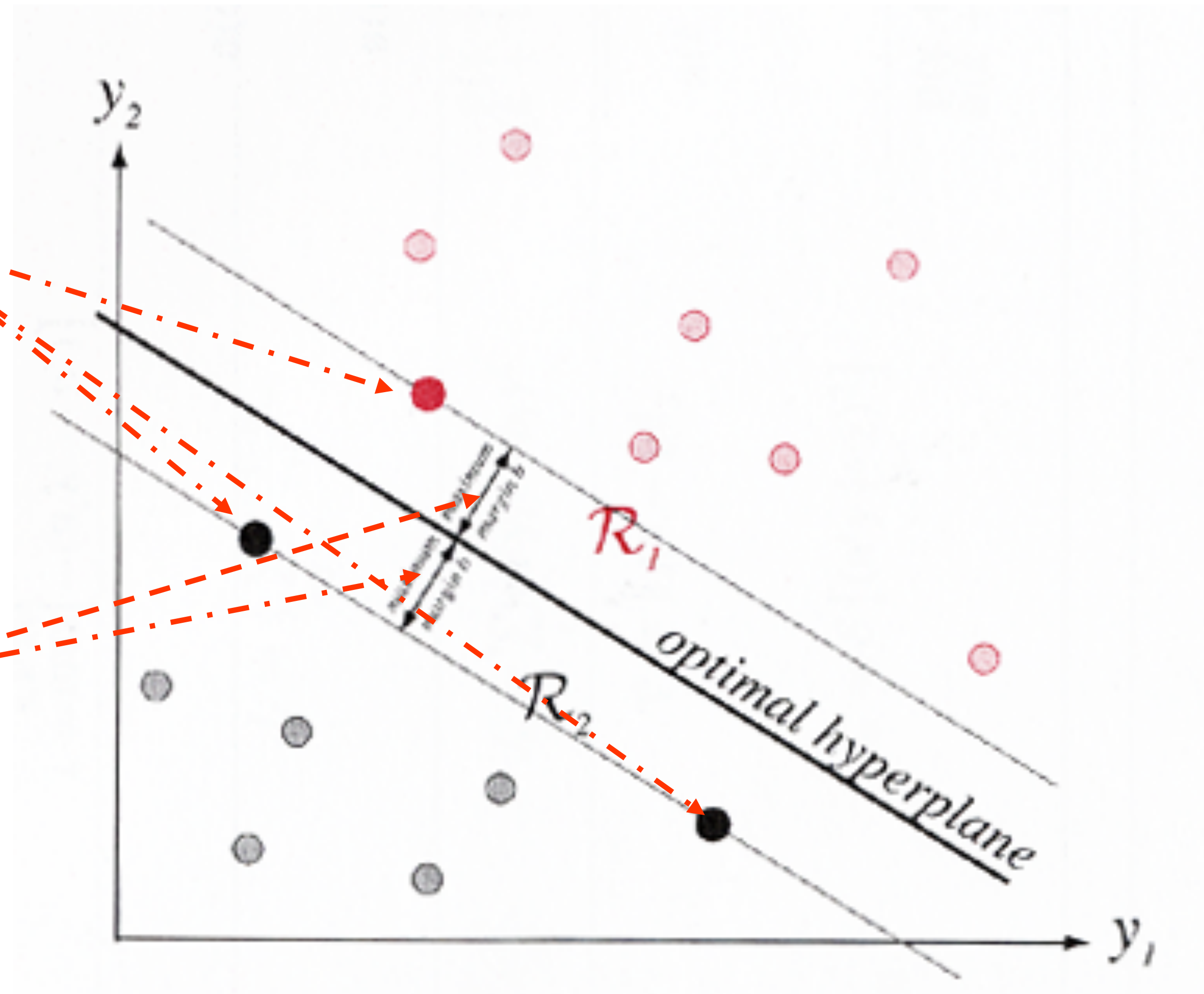
- Margin:
  - Perpendicular distance between between boundary and the closest data point (left)
- Maximizing margin leads to a particular choice of decision boundary
  - Determined by a subset of the data points
    - Which are known as *support vectors* (circles)

Support Vectors and Margin

- Support vectors are those nearest patterns at distance b from hyperplane
- SVM finds hyperplane with maximum distance

  (margin distance b)

  from nearest training patterns



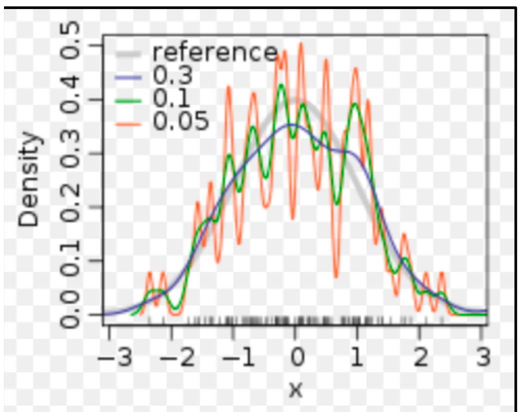
Three support vectors are shown as solid dots

Motivation for maximum margin

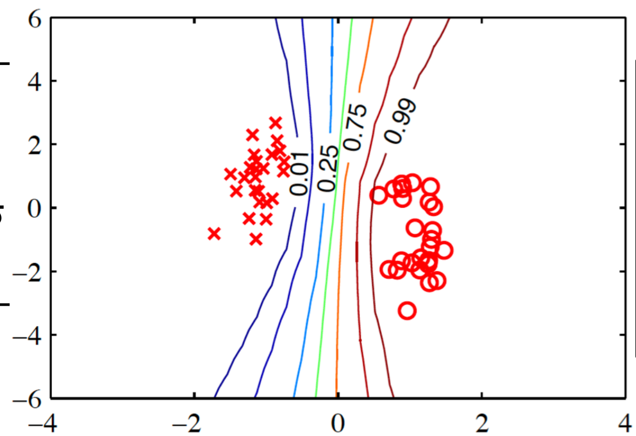
- Maximum margin solution is motivated by computational learning theory
- Simple insight
  - Based on a hybrid of generative and discriminative approaches
    - Seen next (Tong, Koller 2000)

Max Margin Insight

- Model class-conditionals $p(\boldsymbol{x} \mid C_i)$
  - Using Parzen estimator with Gaussian kernels
    - With common parameter σ^2



100 normally distributed nos with decreasing σ $p(\boldsymbol{x}| C_i)$



Different priors $p(C_i)$ for linearly separable data leads to a decision boundary that lies in the middle

- $$P(C_1 \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid C_1)P(C_1)}{P(\boldsymbol{x} \mid C_1)P(C_1) + P(\boldsymbol{x} \mid C_2)P(C_2)}$$ defines the Bayes classifier
  - Determines best hyperplane for learned density model
- As $\sigma^2 \rightarrow 0$ hyperplane has maximum margin
  - In the limit hyperplane becomes independent of points that are not support vectors

Distance of a point $\boldsymbol{x}$ from plane

Theorem: Given a plane $y(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$, perpendicular distance from $\boldsymbol{x}$ to the plane is $\boxed{r = \dfrac{y(\boldsymbol{x})}{||\,\boldsymbol{w}\,||}}$

Proof: $\boxed{\boldsymbol{x} = \boldsymbol{x}_p + r\dfrac{\boldsymbol{w}}{||\,\boldsymbol{w}\,||}}$

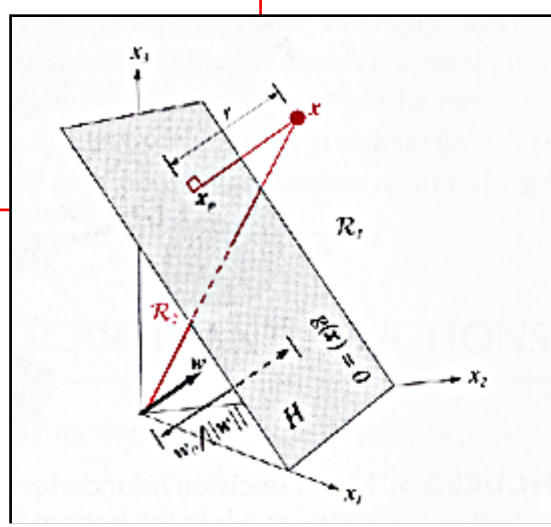
$$y(\boldsymbol{x}) = \boldsymbol{w}^t\left(x_p + r\frac{\boldsymbol{w}}{||\,\boldsymbol{w}\,||}\right) + b = w^t x_p + w_0 + r\frac{w^t w}{||\,w\,||} = \underbrace{y(x_p)}_{=0} + r\frac{||\,w\,||^2}{||\,w\,||} = r\,||\,w\,||$$

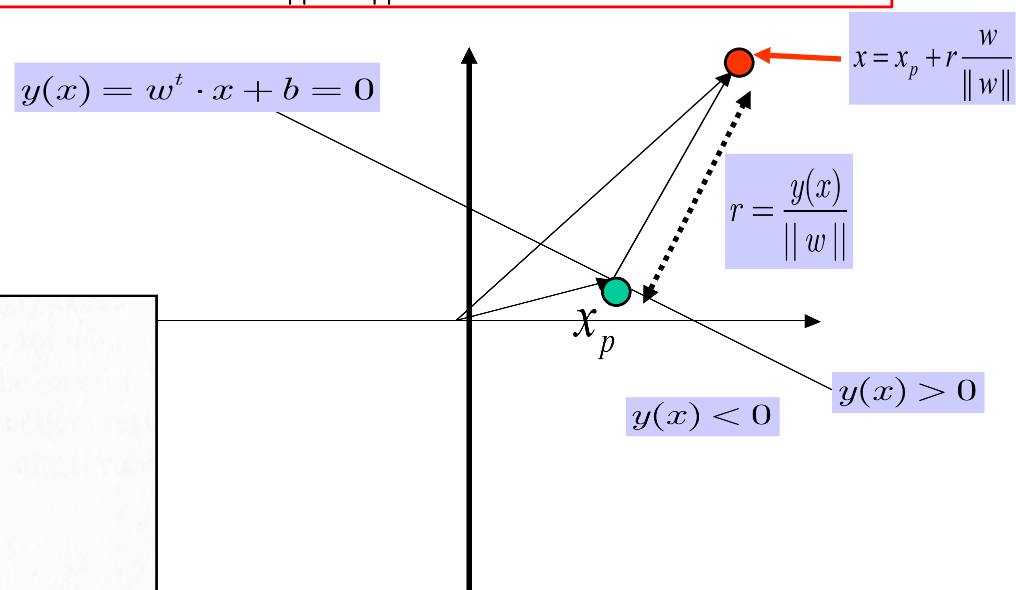
QED

Corollary: Distance of origin to plane is

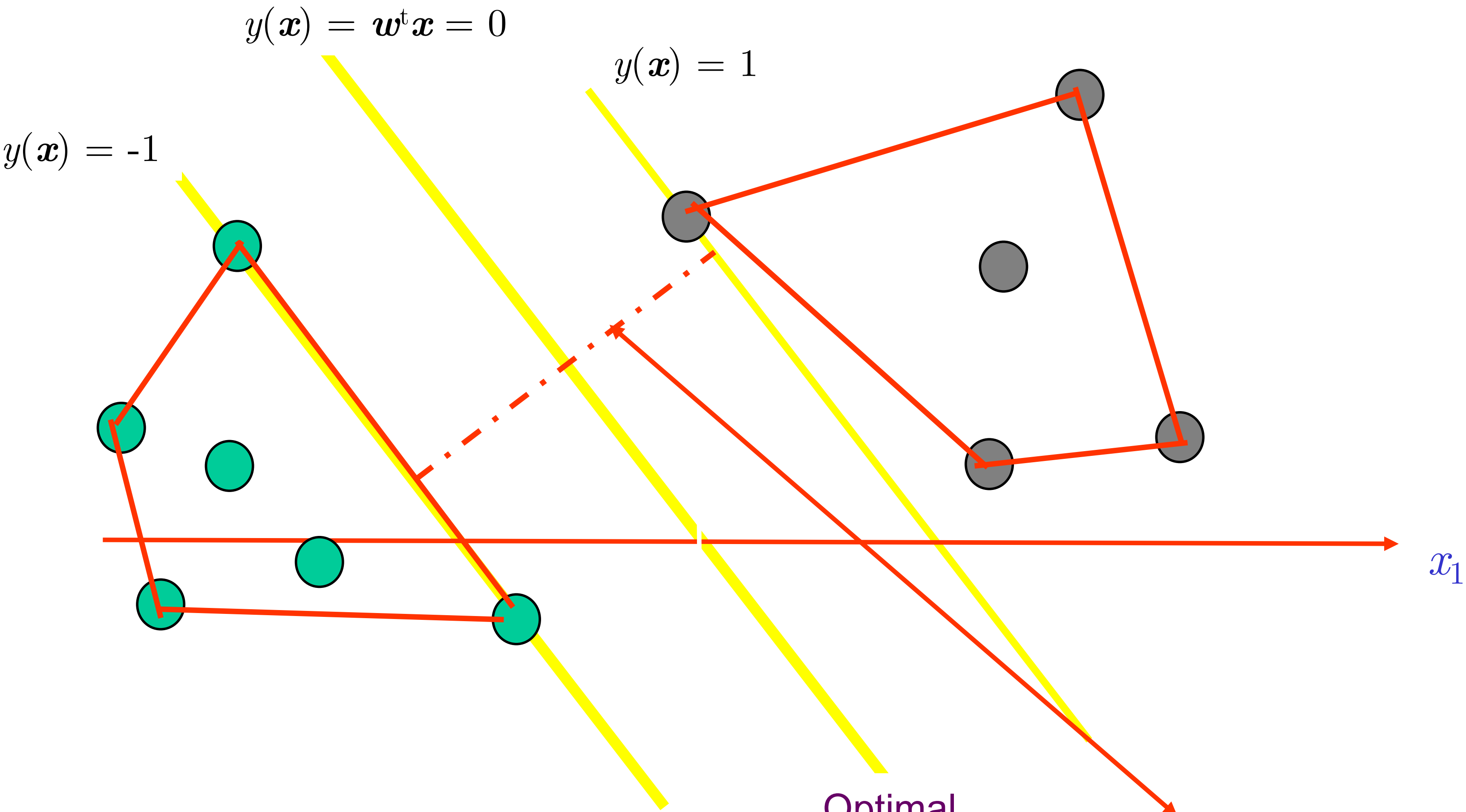
$$r = y(\mathbf{0})/||\boldsymbol{w}|| = b/||\boldsymbol{w}||$$

Since $y(\mathbf{0})=\boldsymbol{w}^{\mathrm{T}}\mathbf{0}+b=b$ then $b=0$ implies that plane passes through origin





SVM Margin geometry



Optimal hyperplane is orthogonal to.. → Shortest line connecting Convex hulls, which has length = $2/||\boldsymbol{w}||$

Maximum Margin Solution

- We interested in solutions for which

  $t_n y(\boldsymbol{x}_n) > 0$ for all n

  - Distance of point $\boldsymbol{x}_n$ to decision surface is

$$\frac{t_n y(\boldsymbol{x}_n)}{||\boldsymbol{w}||} = \frac{t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b)}{||\boldsymbol{w}||}$$

  - Margin is closest point $\boldsymbol{x}_n$ of data set
    - We wish to optimize $\boldsymbol{w}$, b to maximize distance
- Maximum margin solution is found from

$$\boxed{\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{||\boldsymbol{w}||} \min_n \left[t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b) \right] \right\}}$$

- Direct solution is very complex
  - So convert into equivalent problem

Equivalent problem

- If we rescale $\boldsymbol{w} \rightarrow \kappa \boldsymbol{w}$ and $b \rightarrow \kappa b$ then the distance from any point $\boldsymbol{x}_n$ to the decision surface given by $t_n y(\boldsymbol{x}_n)/||\boldsymbol{w}||$ is unchanged
  - We can use this freedom to set, for the point closest to the surface, $\boxed{t_n\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)+b\right)=1}$
    - In this case all points will satisfy

$$\boxed{t_n\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)+b\right)\geq 1 \quad n=1,..,N}$$

  - This is the *canonical representation* of the hyperplane

Canonical Representation

$$t_n\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)+b\right)\geq 1 \quad n=1,..,N$$

- When equality holds, constraints are *active*
  - For the remainder they are *inactive*
  - There will always be at least one active constraint, because there will be a closest point
  - Once margin is maximized there will be two active constraints

Optimization Problem (OP) Restated

- Augmented space:

  $y(\boldsymbol{x}) = \boldsymbol{w}^t \boldsymbol{x}$ by choosing $w_0 = b$ and $x_0=1$, i.e, plane passes through origin

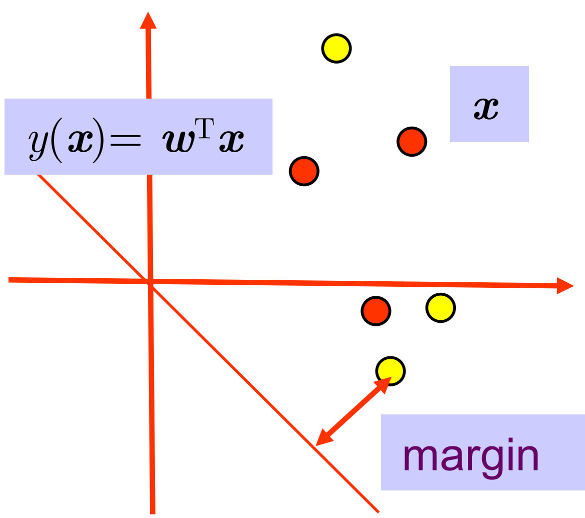
  - Canonical representation of decision hyperplane is then

    $$t_n \; y(\boldsymbol{x}_n) \geq 0, \; n = 1,.., N$$

  - Since distance of a point y to hyperplane $y(\boldsymbol{x})=0$ is $\boxed{\dfrac{y(\boldsymbol{x})}{||\,\boldsymbol{w}\,||}}$

    Maximizing the margin is equivalent to maximizing $||\boldsymbol{w}||^{-1}$

    Which is equivalent to minimizing $||\boldsymbol{w}||^2$



So the optimization problem

$$\boxed{\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{||\,\boldsymbol{w}\,||} \min_n \left[t_n (\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b) \right] \right\}}$$

is equivalent to

$$\boxed{\arg\min_{\boldsymbol{w},b} \frac{1}{2} \left\| \boldsymbol{w} \right\|^2}$$

The factor ½ is for later convenience

subject to the constraints

$$t_n \; (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) + b) \geq 1 \quad n=1,..,N$$

This is an example of quadratic programming

We are trying to maximize a quadratic function subject to a set of linear inequality constraints

SVM Constrained Optimization

$$\boxed{\begin{array}{l}\text{Optimize} \\ \arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to constraints} \\ t_n(\boldsymbol{w}^t\phi(\boldsymbol{x}_n)+b)\geq 1, \quad n=1,\ldots..N\end{array}}$$

- Can be cast as an unconstrained problem
- by introducing Lagrange undetermined multipliers with one multiplier $a_n \geq 0$ for each constraint
- The Lagrange function we wish to minimize is

$$\boxed{L(\boldsymbol{w},b,\boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} a_n\left[t_n(\boldsymbol{w}^t\phi(\boldsymbol{x}_n)+b)-1\right]}$$

where $\boldsymbol{a}=(a_1,..,a_N)^{\mathrm{T}}$

Note the minus sign in front of the Lagrange multiplier term
Because we are minimizing wrt $\boldsymbol{w}$ and b, and maximizing wrt $\boldsymbol{a}$

Optimization of Lagrange function

- We wish to minimize

$$\boxed{L(\boldsymbol{w},b,\boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} a_n \left[t_n \boldsymbol{w}^t (\phi(\boldsymbol{x}_n) + b) - 1 \right]}$$

- We seek to minimize $L(\ )$
  - with respect to $\boldsymbol{w}$ and b
  - maximize it w.r.t . the undetermined multipliers $a_n \geq 0$
- Last term represents the goal of classifying the points correctly
- Karush-Kuhn-Tucker construction shows that this can be recast as a maximization problem which is computationally better

Eliminating $\boldsymbol{w}$ and b

$$L(\boldsymbol{w},b,\boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} a_n \left[t_n \boldsymbol{w}^t(\phi(\boldsymbol{x}_n) + b) - 1\right]$$

- Setting derivatives of $L(\boldsymbol{w},b,\boldsymbol{a})$ wrt $\boldsymbol{w}$ and b equal to zero
  - We obtain the following conditions

$$\boldsymbol{w} = \sum_n a_n t_n \boldsymbol{x}_n \qquad\qquad 0 = \sum_n a_n t_n$$

  - Eliminating $\boldsymbol{w}$ and b from $L(\boldsymbol{w},b,\boldsymbol{a})$ using these conditions then gives the dual representation of the maximum margin problem

Dual O.P.

- Problem is one of maximizing

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_m a_n t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

  - Subject to the constraints

$$\sum_{n=1}^{N} a_n t_n = 0 \qquad a_n \geq 0, \quad n = 1, \ldots.., N$$

  given the training data

- Here the kernel function is defined by

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^t \cdot \phi(\boldsymbol{x}')$$

- Again a quadratic optimization problem
  - Optimize a quadratic function subject to a set of inequality constraints

Complexity of Primal and Dual

- Quadratic programming complexity is $O(M^3)$
  - Primal OP of minimizing
  - $\boxed{\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2}$ over M basis functions
  - Dual OP of maximizing
  - $\boxed{\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_m a_n t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)}$ over N data points
- For $M << N$ it appears disadvantageous
  - But it allows model to be reformulated using kernels
    - So it can be applied to feature spaces whose dimensionality M exceeds no of data points N
    - Kernel formulation also makes clear that $k(\boldsymbol{x},\boldsymbol{x}')$ be positive definite, which bounds $\mathrm{L}(\boldsymbol{a})$ is lower-bounded

Classifying new points

- To classify points using trained model:
  - Evaluate sign of $y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b$
    - This can be expressed in terms of the parameters $\{a_n\}$ and the kernel function by substituting for $\boldsymbol{w}$ using $\boxed{\boldsymbol{w} = \sum_n a_n t_n \boldsymbol{x}_n}$ to give

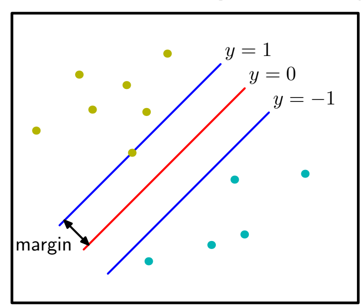
$$\boxed{y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b}$$

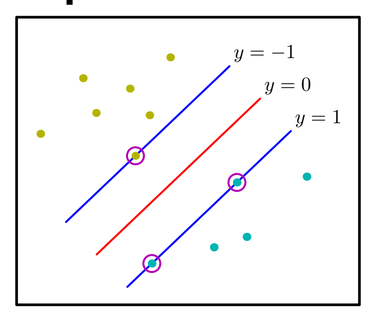
Definition of Support Vectors

- This OP satisfies KKT conditions
  - which requires the following hold:

$$a_n \geq 0$$
$$t_n y(\boldsymbol{x}_n) - 1 \geq 0$$
$$a_n \{t_n y(\boldsymbol{x}_n) - 1\} = 0$$

  - Thus for every data point either $a_n = 0$ or $t_n y(\boldsymbol{x}_n) = 1$
    - A point with $a_n = 0$ will not appear in sum $y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b$
  - Remaining data points are called support vectors
    - Because they satisfy $t_n y(\boldsymbol{x}_n) = 1$ they correspond to points that lie on the maximum margin hyperplanes in feature space





This property is central to practical applicability of SVMs. Once it is trained a significant proportion of Data points can be discarded and only the support vectors retained

Solution for b

- Having solved the quadratic programming problem and found a we can can determine value of threshold parameter b
  - Noting that for any support vector $t_n y(\boldsymbol{x}_n) = 1$
  - Using $\boxed{y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b}$ gives $\boxed{t_n \left(\sum_{m \in S} a_m t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) + b \right) = 1}$
    - Where S denotes the indices of support vectors
  - Solving for b gives $\boxed{b = \frac{1}{N_S} \left(t_n - \sum_{m \in S} a_m t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) \right)}$
    - Where N_S is the total no of support vectors

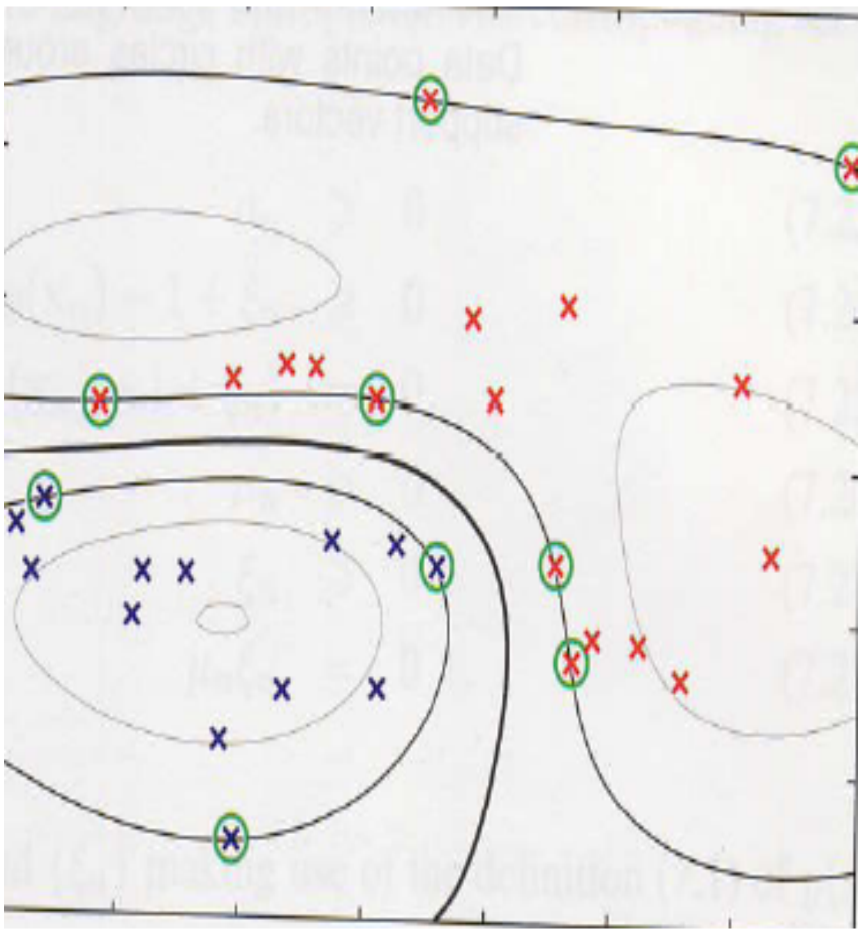
Equivalent Error Function of SVM

- For comparison with alternative models
  - Express the maximum margin classifier in terms of minimization of an error function:

$$\boxed{\sum_{n=1}^{N} E_{\infty}\left(y(\boldsymbol{x}_n)t_n - 1\right) + \lambda \left\|\boldsymbol{w}\right\|^2}$$

    - Where $E_\infty(z)$ is zero if $z \geq 0$ and ∞ otherwise
      – Ensures that $t_n\ (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)+b) \geq 1 \quad n=1,..,N$ are satisfied
    - Gives infinite error if point is misclassified
    - Zero error if it was classified correctly
  - We optimize parameters to maximize margin

Example of SVM Classifier



- Two classes in two dimensions
- Synthetic Data
- Shows contours of constant $y(\boldsymbol{x})$
- Obtained from SVM with Gaussian kernel function
- Decision boundary is shown
- Margin boundaries are also shown
- Support vectors are shown
- Shows sparsity of SVM

Summary of SVM Optimization Problems

Different Notation here!

Dual Optimization Problem

Primal OP: minimize $P(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w} \cdot \vec{w} + C\sum_{i=1}^{n} \xi_i$

s. t. $y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i$ *and* $\xi_i \geq 0$

Lemma: The solution $w°$ can always be written as a linear combination

$$\vec{w}° = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i \qquad \alpha_i \geq 0$$

of the training data.

Quadratic term

Dual OP: maximize $D(\vec{\alpha}) = \left(\sum_{i=1}^{n} \alpha_i\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$

s.t. $\sum_{i=1}^{n} \alpha_i y_i = 0 \qquad$ *and* $\qquad 0 \leq \alpha_i \leq C$

==> positive semi-definite quadratic program

SVM with Kernels

Training: maximize $D(\vec{\alpha}) = \left(\sum_{i=1}^{n} \alpha_i\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j)$

$$\text{s. t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad und \qquad 0 \le \alpha_i \le C$$

Classification: For new example x $\quad h(\vec{x}) = sign\left(\sum_{x_i \in SV} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b\right)$

New hypotheses spaces through new Kernels:

Linear: $K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$

Polynomial: $K(\vec{x}_i, \vec{x}_j) = [\vec{x}_i \cdot \vec{x}_j + 1]^d$

Radial Basis Functions: $K(\vec{x}_i, \vec{x}_j) = \exp(-|\vec{x}_i - \vec{x}_j|^2 / \sigma^2)$

Sigmoid: $K(\vec{x}_i, \vec{x}_j) = \tanh(\gamma(\vec{x}_i - \vec{x}_j) + c)$

Kernel Function: key property

- If kernel function is chosen with property

  $$k(\boldsymbol{x},\boldsymbol{x}') = (\phi(\boldsymbol{x})^{\mathrm{T}}.\phi(\boldsymbol{x}'))$$

  then computational expense of increased dimensionality is avoided.

- Polynomial kernel $k(\boldsymbol{x},\boldsymbol{x}') = (\boldsymbol{x}^T.\boldsymbol{x}')^p$ can be shown (next slide) to correspond to a map ϕ into the space spanned by **all** products of exactly d dimensions.

A Polynomial Kernel Function

Suppose $\boldsymbol{x} = (x_1, x_2)$ is a 2 - dimensional input vector

The feature space is 3 - dimensional :

$$\phi(\boldsymbol{x}) = \left(x_1^{\,2}, \sqrt{2}x_1x_2, x_2^{\,2}\right)^t$$

Then inner product is

$$\boxed{\phi(\boldsymbol{x})^T \phi(\boldsymbol{y}) = \left(x_1^{\,2}, \sqrt{2}x_1x_2, x_2^{\,2}\right)^t \left(y_1^{\,2}, \sqrt{2}y_1y_2, y_2^{\,2}\right) = (x_1y_1 + x_2y_2)^2}$$

same

Polynomial kernel function to compute the same value is

$$\boxed{k(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}^T.\boldsymbol{y})^2 = \left(\left(x_1, x_2\right)^T \left(y_1, y_2\right)\right)^2 = (x_1y_1 + x_2y_2)^2}$$

or $k(\boldsymbol{x},\boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$

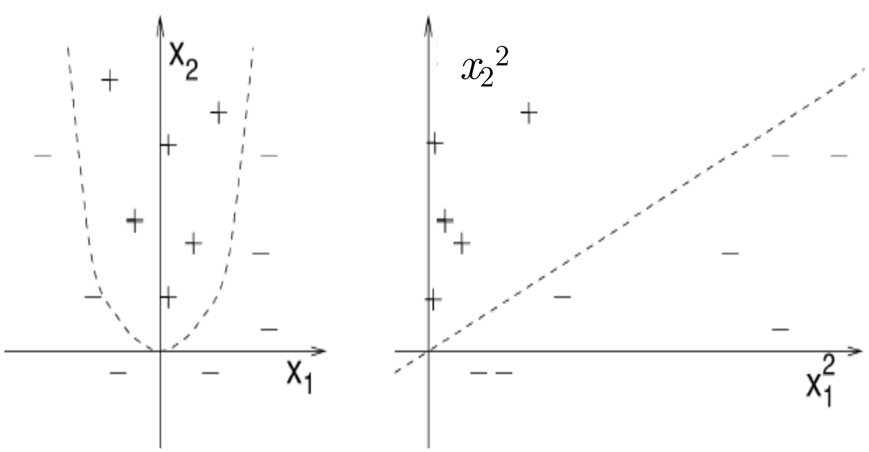
- Inner product $\phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$ needs computing six feature values and $3 \times 3 = 9$ multiplications
- Kernel function $k(\boldsymbol{x},\boldsymbol{y})$ has 2 multiplications, one add and a squaring

Another Polynomial kernel function

Example

Input Space: $\vec{x} = (x_1, x_2)$ (2 Attributes)

Feature Space: $\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$ (6 Attributes)



- $k(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}.\boldsymbol{y} + 1)^2$
- This one maps $d = 2,\ p = 2$ into a *six*- dimensional space
- Contains all the powers of x

$$k(\boldsymbol{x},\boldsymbol{y}) = \phi(\boldsymbol{x})\phi(\boldsymbol{y})$$

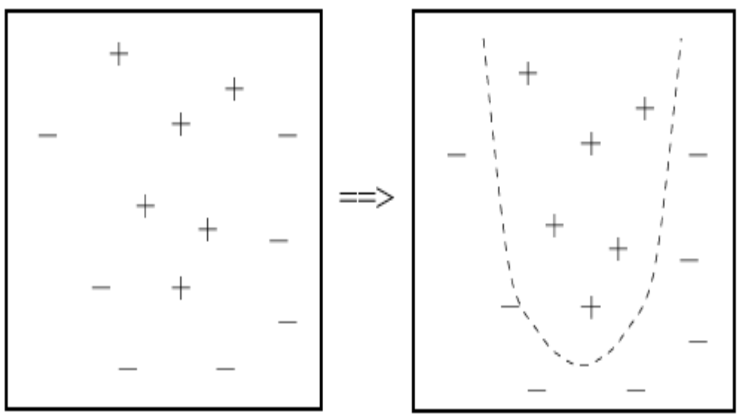
where

$$\phi(\boldsymbol{x}) = \left(x_1^{\,2}, x_2^{\,2}, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1\right)$$

- Inner product needs 36 multiplications
- Kernel function needs 4 multiplications

Non-Linear Case

Non-Linear Problems



Problem:

- some tasks have non-linear structure
- no hyperplane is sufficiently accurate

How can SVMs learn non-linear classification rules?

- Mapping function $\phi\,(.)$ to a sufficiently high dimension
- So that data from two categories can always be separated by a hyperplane
- Assume each pattern x_k has been transformed to

  $\phi(x_k)$, for $k=1,\ldots,\ n$
- First choose the non-linear ϕ functions
  - To map the input vector to a higher dimensional feature space
- Dimensionality of space can be arbitrarily high only limited by computational resources

Mapping into Higher Dimensional Feature Space

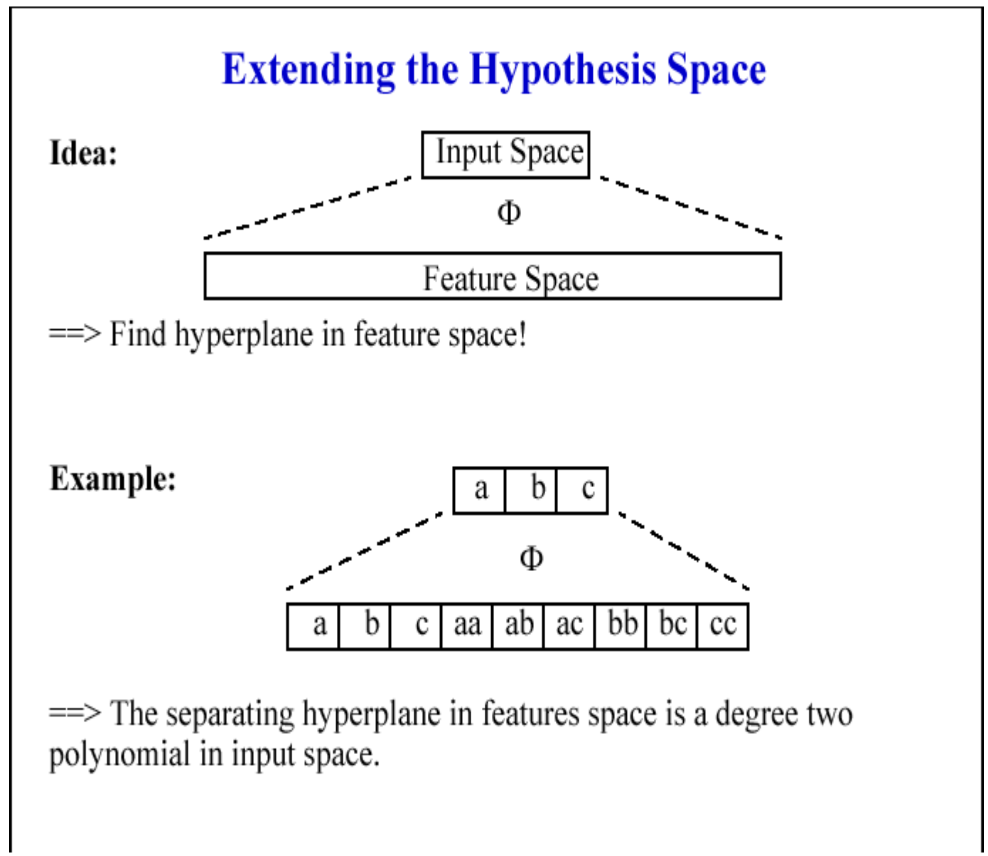
- Mapping each input point $\boldsymbol{x}$ by map

$$\Phi(\boldsymbol{x}) = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}$$

Points on 1-d line are mapped onto curve in 3-d.

- Linear separation in 3-d space is possible. Linear discriminant function in 3-d is in the form

$$y(\boldsymbol{x}) = a_1 x_1 + a_2 x_2 + a_3 x_3$$

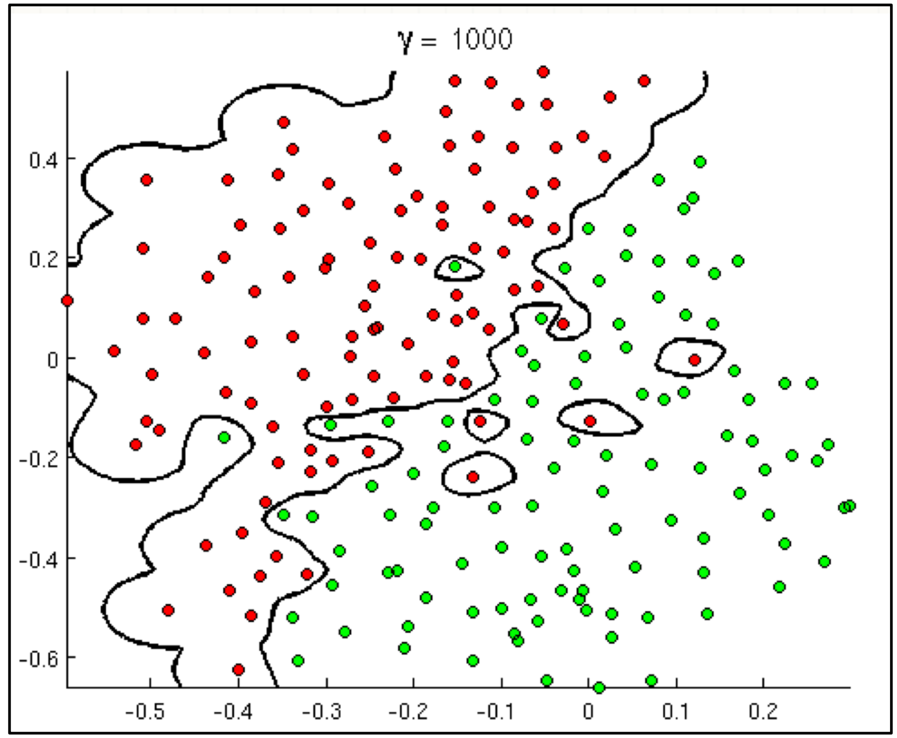


Gaussian (RBF) Kernel

- $\phi(\boldsymbol{x})$ maps $\boldsymbol{x}$ to an infinite space

$$K(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right) = \phi(x)^T\phi(y)$$

$$\phi(x) = e^{-x^2/2\sigma^2}\left[1, \sqrt{\frac{1}{1!\sigma^2}}x, \sqrt{\frac{1}{2!\sigma^4}}x^2, \sqrt{\frac{1}{3!\sigma^6}}x^3, \ldots\right]^T$$



- Achieves perfect separation

For sufficiently small kernel bandwidth, decision boundary will look like you just drew little circles around the points whenever they are needed to separatepositive and negative examples.

RBF Features from Taylor's Series

Can be inner product in infinite dimensional space

Assume $x \in R^1$ and $\gamma > 0$.

$$e^{-\gamma\|x_i - x_j\|^2} = e^{-\gamma(x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2}$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \cdots\right)$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}}x_i \cdot \sqrt{\frac{2\gamma}{1!}}x_j + \sqrt{\frac{(2\gamma)^2}{2!}}x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}}x_j^2 + \sqrt{\frac{(2\gamma)^3}{3!}}x_i^3 \cdot \sqrt{\frac{(2\gamma)^3}{3!}}x_j^3 + \cdots\right) = \phi(x_i)^T \phi(x_j),$$

where

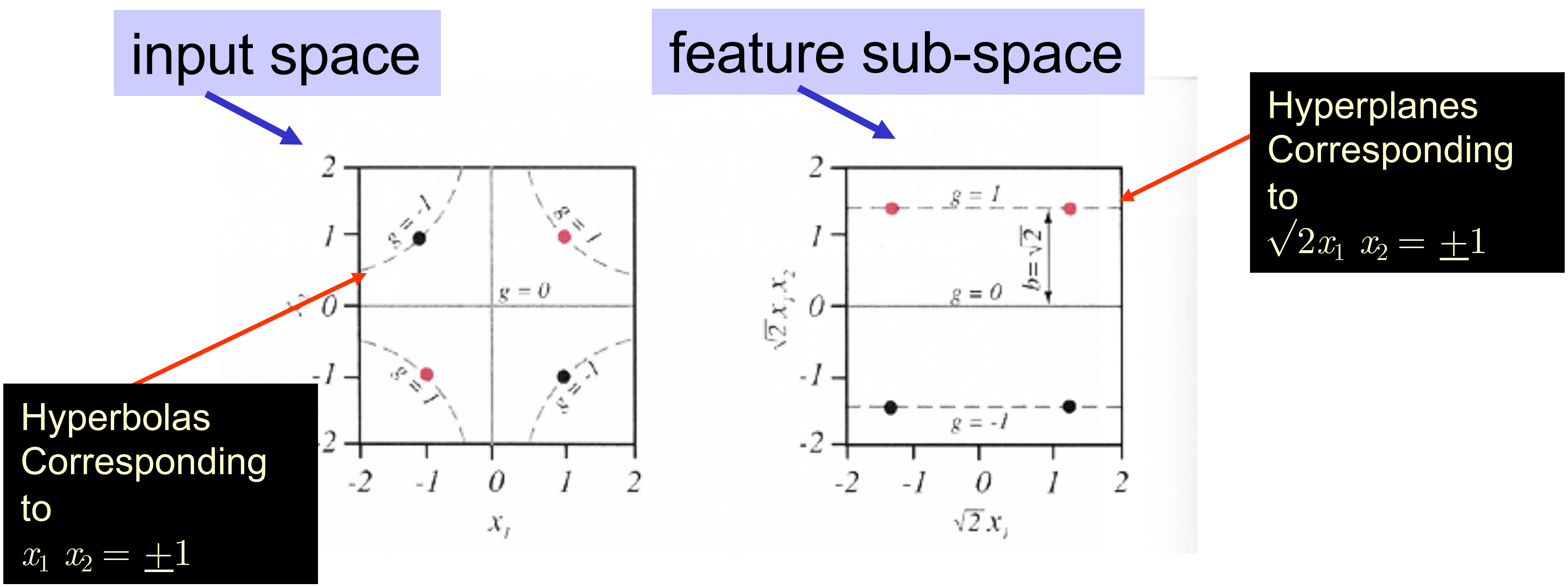
$$\phi(x) = e^{-\gamma x^2}\left[1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \sqrt{\frac{(2\gamma)^3}{3!}}x^3, \cdots\right]^T.$$

Pattern Transformation using Kernels

- Problem with high-dimensional mapping
  - Very many parameters
  - Polynomial of degree p over d variables leads to $\mathrm{O}(d^p)$ variables in feature space
    - Example: if $d = 50$ and $p = 2$ we need a feature space of size 2500
- Solution:
  - Dual Optimization problem needs only inner products
  - Each pattern $\boldsymbol{x}_k$ transformed into $\boldsymbol{\phi}(\boldsymbol{x}_k)$
  - Dimensionality of mapped space can be arbitrarily high

SVM for the XOR problem

- XOR: binary valued features $x_1,\ x_2$
- Not solved by linear discriminant
- Function ϕ maps input $\boldsymbol{x} = [x_1,\ x_2]$ into six-D feature space $\boldsymbol{y} = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2\ ,\ x_1{}^2\ ,\ x_2{}^2]^{\mathrm{T}}$



SVM for XOR: maximization problem

- We seek to maximize

$$\sum_{k=1}^{4} a_k - \frac{1}{2}\sum_{k=1}^{4}\sum_{j=1}^{4} a_k a_j t_j t_k \phi(\boldsymbol{x}_j)\phi(\boldsymbol{x}_k)$$

- Subject to the constraints $\boxed{0 = \sum_n a_n t_n}$, i.e.,

$$\boxed{\begin{array}{l} a_1 - a_2 + a_3 - a_4 = 0 \\ 0 \le a_k \qquad k = 1,\ 2,\ 3,\ 4 \end{array}}$$

- From problem symmetry, at the solution

$a_1 = a_3$ and $a_2 = a_4$

SVM for XOR: maximization problem

- Can use iterative gradient descent
- Or use analytical techniques for small problem
- The solution is

$$\boldsymbol{a}^* = (1/8,\ 1/8,\ 1/8,\ 1/8)$$

- Last term of Optimizn Problem implies that all four points are support vectors (unusual and due to symmetric nature of XOR)
- The final discriminant function is $y(x_1,x_2) = x_1 \times x_2$
- Decision hyperplane is defined by $y(x_1,x_2) = 0$
- Margin is given by $b=1/||\boldsymbol{a}|| = \sqrt{2}$

Hyperplanes Corresponding to

$\sqrt{2}x_1\ x_2 = \pm 1$

Hyperbolas Corresponding to

$x_1\ x_2 = \pm 1$