CSC411 Fall 2015

Introduction to Machine Learning

Support Vector Machines & Kernels

Slides by Rich Zemel

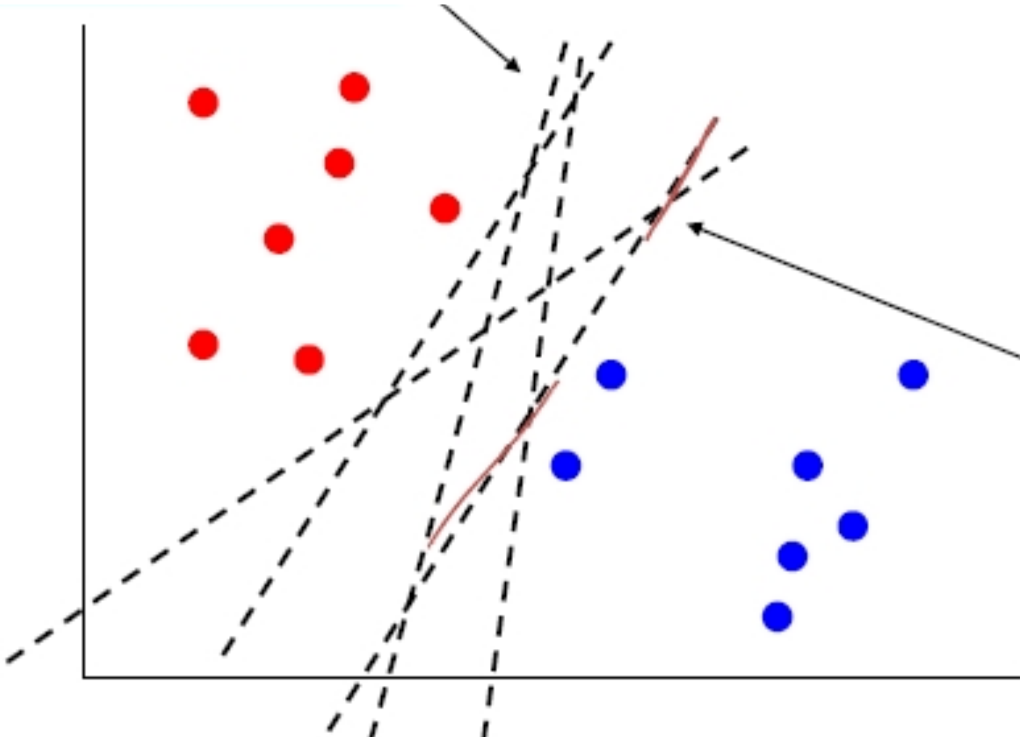
Logistic Regression

Recall logistic regression classifiers

$$\max_{\mathbf{w}} \sum_{i=1}^{n} \ln(1+\exp(\mathbf{w}^T \mathbf{x}_i y_i))$$

Goes over all training points x

Many possible classifiers



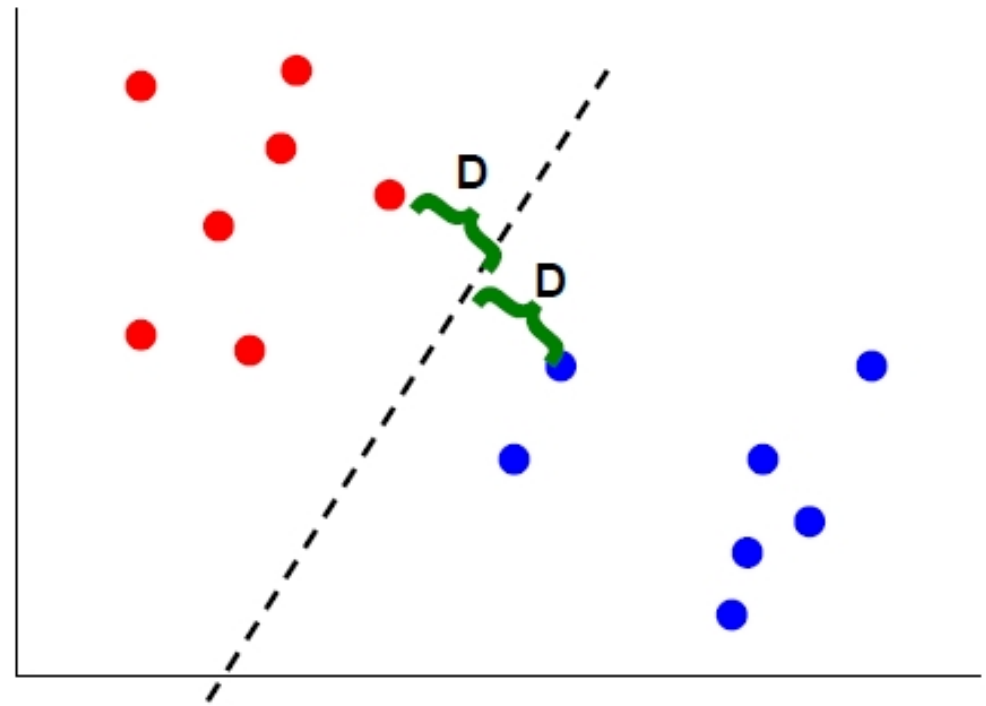
Line closer to the blue nodes since many of them are far away from the boundary

$$y = \begin{cases} +1 \text{ [red] if } \operatorname{sign}(\mathbf{w}^T \mathbf{x} + b \geq 0) \\ -1 \text{ [blue] if } \operatorname{sign}(\mathbf{w}^T \mathbf{x} + b < 0) \end{cases}$$

Max margin classification

Instead of fitting all the points, focus on boundary points

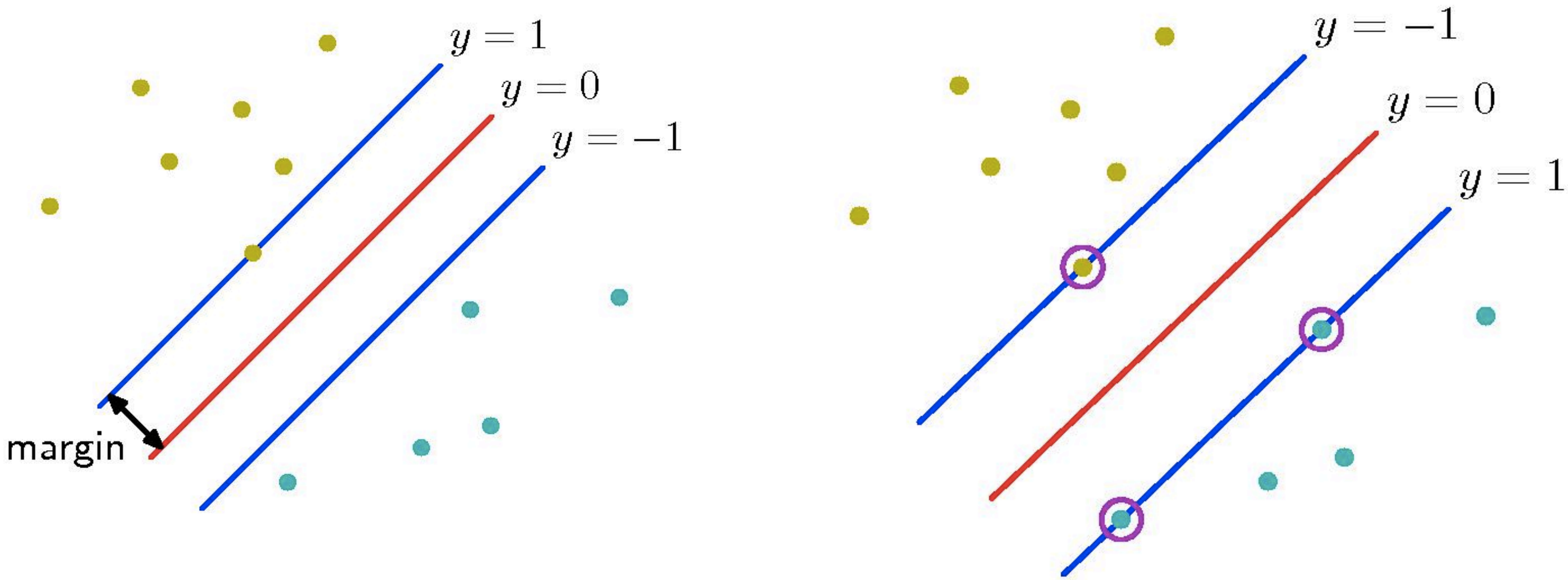
Aim: learn a boundary that leads to the largest margin (buffer) from points on both sides



Why: intuition; theoretical support; and works well in practice

Subset of vectors that support (determine boundary) are called the support vectors

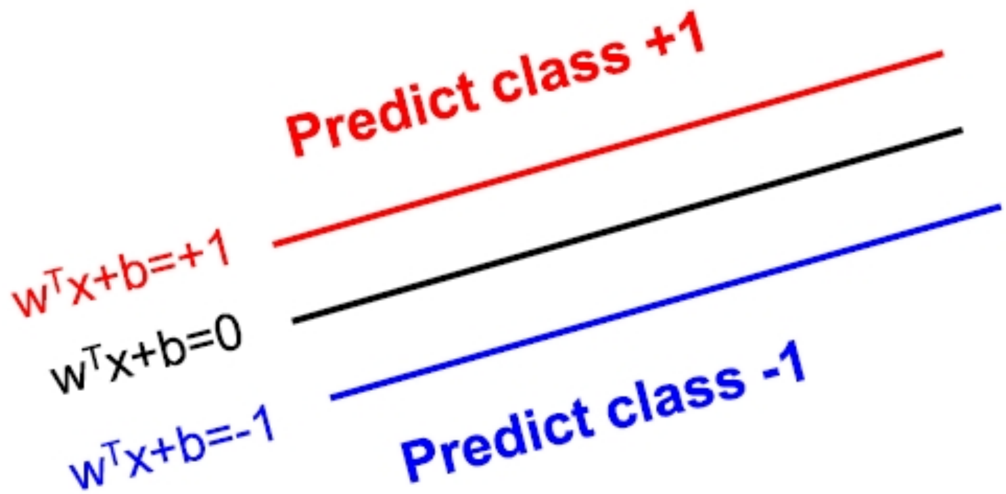
Max margin classification



- The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points.
- Maximizing the margin leads to a particular choice of decision boundary.

Linear SVM

Max margin classifier: inputs in margin are of unknown class



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

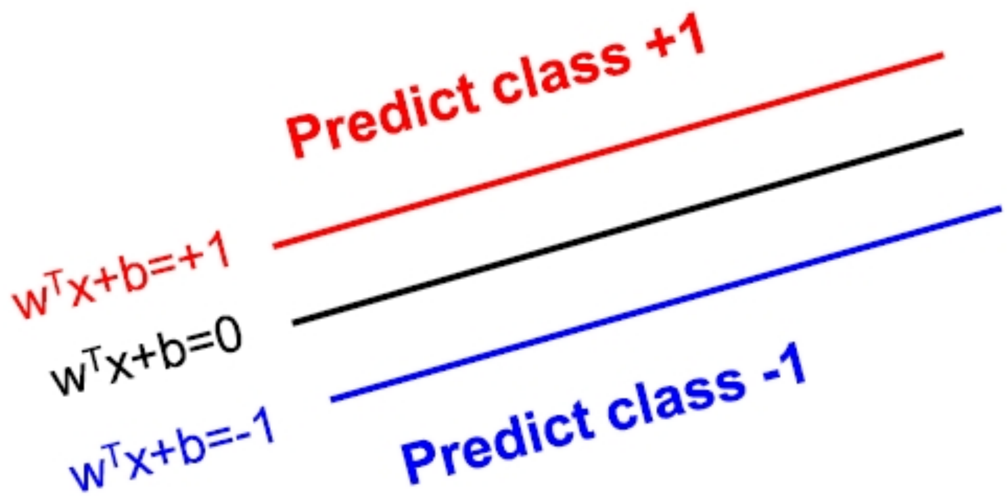
First note that the **w** vector is orthogonal to the +1 plane

if **u** and **v** are two points on that plane, then $\mathbf{w}^T(\mathbf{u}-\mathbf{v}) = 0$

Same is true for -1 plane

Also: for point **x+** on +1 plane and **x-** nearest point on -1 plane:

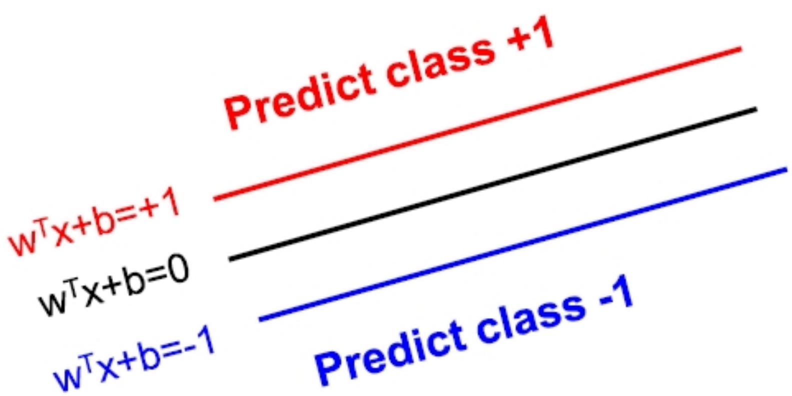
$\mathbf{x+} = \lambda\mathbf{w} + \mathbf{x-}$



Computing the Margin

Also: for point **x+** on +1 plane and **x-** nearest point on -1 plane:

x+ = λ**w** + **x-**



$$\mathbf{w}^T\mathbf{x}^+ + b = 1$$

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

$$\mathbf{w}^T\mathbf{x}^- + b + \lambda\mathbf{w}^T\mathbf{w} = 1$$

$$-1 + \lambda\mathbf{w}^T\mathbf{w} = 1$$

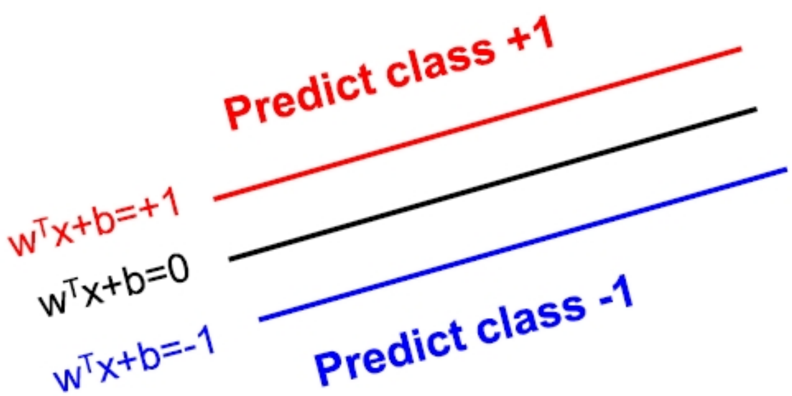
$$\lambda = 2 / \mathbf{w}^T\mathbf{w}$$

Computing the Margin

Define the margin M to be the distance between the +1 and -1 planes

We can now express this in terms of **w** →

to maximize the margin we minimize the length of **w**



$$M = \left\| \mathbf{x}^{+} - \mathbf{x}^{-} \right\|$$

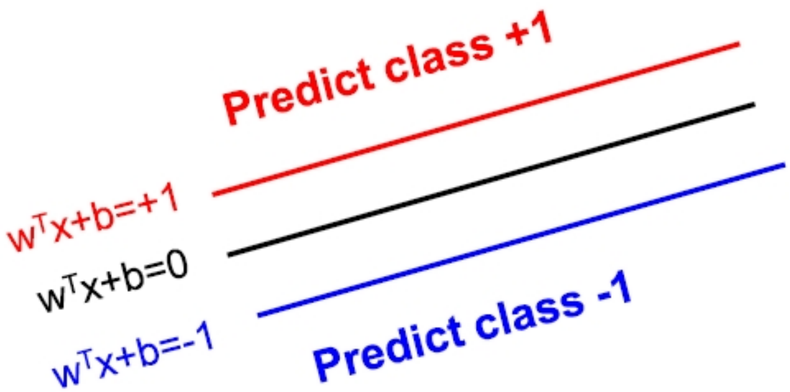
$$= \left\| \lambda \mathbf{w} \right\| = \lambda \sqrt{\mathbf{w}^T \mathbf{w}}$$

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

Learning a Margin-Based Classifier

We can search for the optimal parameters (**w** and b) by finding a solution that:

1. Correctly classifies the training examples: $\{x_i, y_i\}$, i=1,..,n
2. Maximizes the margin (same as minimizing **$\mathbf{w}^T\mathbf{w}$)**



$$\min \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t.\ (\mathbf{w}^T\mathbf{x}_i + b)y_i \geq 1 \forall i$$

Can optimize via gradient descent, EM, etc.

Apply Lagrange multipliers: formulate equivalent problem

Learning a Linear SVM

Convert the constrained minimization to an unconstrained optimization problem: represent constraints as penalty terms:

$$\min \ \frac{1}{2}\|\mathbf{w}\|^2 + \text{penalty}$$

For data $\{(x_i,y_i)\}$ use the following penalty term:

$$\begin{Bmatrix} 0 & \text{if } (\mathbf{w}^T\mathbf{x}_i + b)y_i \geq 1 \\ \infty & \text{otherwise} \end{Bmatrix} = \max_{\alpha_i \geq 0} \ \alpha_i[1-(\mathbf{w}^T\mathbf{x}_i + b)y_i]$$

Rewrite the minimization problem:

$$\min_{\mathbf{w},b}\{\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}\max_{\alpha_i \geq 0}\alpha_i[1-(\mathbf{w}^T\mathbf{x}_i + b)y_i]\}$$

Where $\{\alpha_i\}$ are the Lagrange multipliers

$$= \min_{\mathbf{w},b}\max_{\alpha_i \geq 0}\{\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}\alpha_i[1-(\mathbf{w}^T\mathbf{x}_i + b)y_i]\}$$

Solution to Linear SVM

Swap the 'max' and 'min':

$$\max_{\alpha_i \geq 0} \min_{\mathbf{w},b} \{\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i [1 - (\mathbf{w}^T \mathbf{x}_i + b) y_i]\}$$

$$= \max_{\alpha_i \geq 0} \min_{\mathbf{w},b} J(\mathbf{w}, b; \alpha)$$

First minimize J w.r.t. {**w**,b} for any fixed setting of the Lagrange multipliers:

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}, b; \alpha) = \mathbf{w} - \sum_{i=1}^{n} \alpha_i \mathbf{x}_i y_i = 0$$

$$\frac{\partial}{\partial b} J(\mathbf{w}, b; \alpha) = -\sum_{i=1}^{n} \alpha_i y_i = 0$$

Then substitute back to get final optimization:

$$L = \max_{\alpha_i \geq 0} \{\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)\}$$

Summary of Linear SVM

- Binary and linear separable classification
- Linear classifier with maximal margin
- Training SVM by maximizing $$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$
- Subject to $\alpha_i \geq 0; \sum_{i=1}^{n} \alpha_i y_i = 0$
- Weights: $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$
- Only a small subset of α_i's will be nonzero, and the corresponding x_i's are the support vectors **S**
- Prediction on a new example:

$$y = \text{sign}[b + \mathbf{x} \cdot (\sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i)] = \text{sign}[b + \mathbf{x} \cdot (\sum_{i \in \mathbf{S}} y_i \alpha_i \mathbf{x}_i)]$$

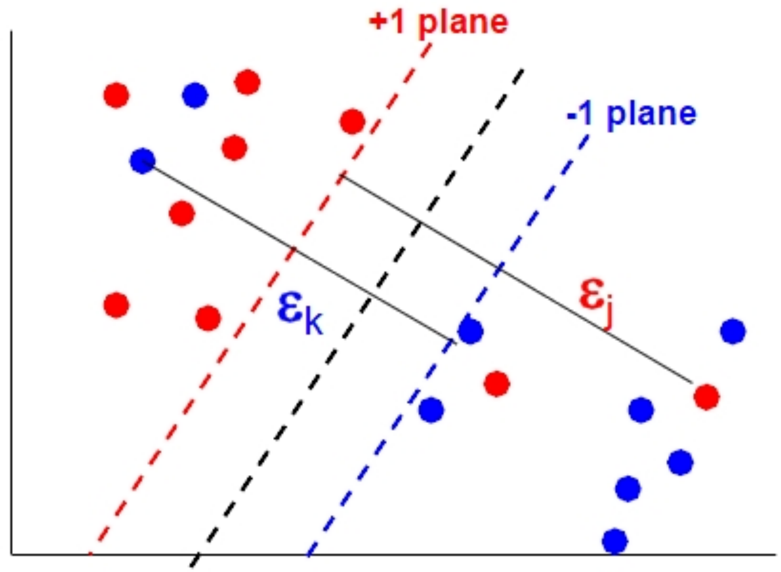
What if data is not linearly separable?

- Introduce slack variables ξ_i

$$\min\left[\frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{i=1}^{n} \xi_i\right]$$

subject to constraints (for all *i*):

$$y_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$



- Example lies on wrong side of hyperplane: $\xi_i > 1 \Rightarrow \sum_i \xi_i$ is upper bound on number of training errors

- lambda trades off training error versus model complexity

- This is known as the soft-margin extension

Non-linear decision boundaries

- Note that both the learning objective and the decision function depend only on dot products between patterns

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \qquad y = \text{sign}[b + \mathbf{x} \cdot (\sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i)]$$

- How to form non-linear decision boundaries in input space?

- Basic idea:

  1. Map data into feature space $\mathbf{x} \rightarrow \phi(\mathbf{x})$

  2. Replace dot products between inputs with feature points

     $$\mathbf{x}_i \cdot \mathbf{x}_j \rightarrow \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$

  3. Find linear decision boundary in feature space

- Problem: what is a good feature function $\varphi(\mathbf{x})$?

Kernel Trick

- Kernel trick: dot-products in feature space can be computed as a kernel function

$$\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$$

- Idea: work directly on **x**, avoid having to compute φ(**x**)

- Example:

$$K(\mathbf{a},\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b})^3 = ((a_1, a_2)\cdot(b_1, b_2))^3$$
$$= (a_1b_1 + a_2b_2)^3$$
$$= a_1{}^3b_1{}^3 + 3a_1{}^2b_1{}^2a_2b_2 + 3a_1b_1a_2{}^2b_2{}^2 + a_2{}^3b_2{}^3$$
$$= (a_1{}^3, \sqrt{3}a_1{}^2a_2, \sqrt{3}a_1a_2{}^2, a_2{}^3)\cdot(b_1{}^3, \sqrt{3}b_1{}^2b_2, \sqrt{3}b_1b_2{}^2, b_2{}^3)$$
$$= \phi(\mathbf{a})\cdot\phi(\mathbf{b})$$

Kernels

Examples of kernels (kernels measure similarity):

1. Polynomial $K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + 1)^2$
2. Gaussian $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2 / 2\sigma^2)$
3. Sigmoid $K(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\kappa(\mathbf{x}_1 \cdot \mathbf{x}_2) + a)$

Each kernel computation corresponds to dot product calculation for particular mapping φ(x): implicitly maps to high-dimensional space

Why is this useful?

1. Rewrite training examples using more complex features
2. Dataset not linearly separable in original space may be linearly separable in higher dimensional space

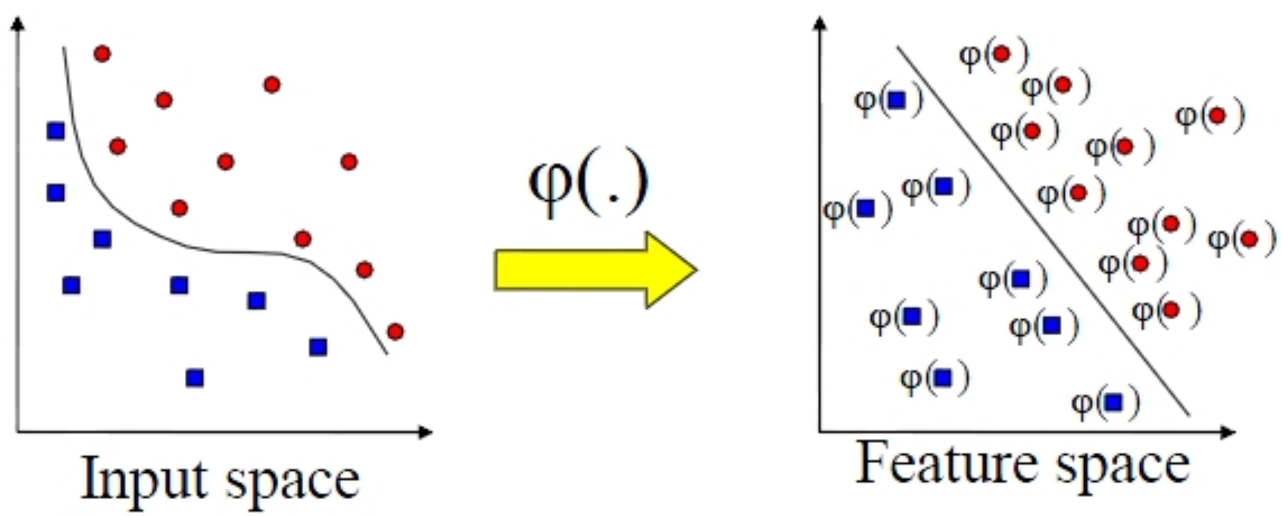
Input transformation

Mapping to a feature space can produce problems:

- High computational burden due to high dimensionality
- Many more parameters

SVM solves these two issues simultaneously

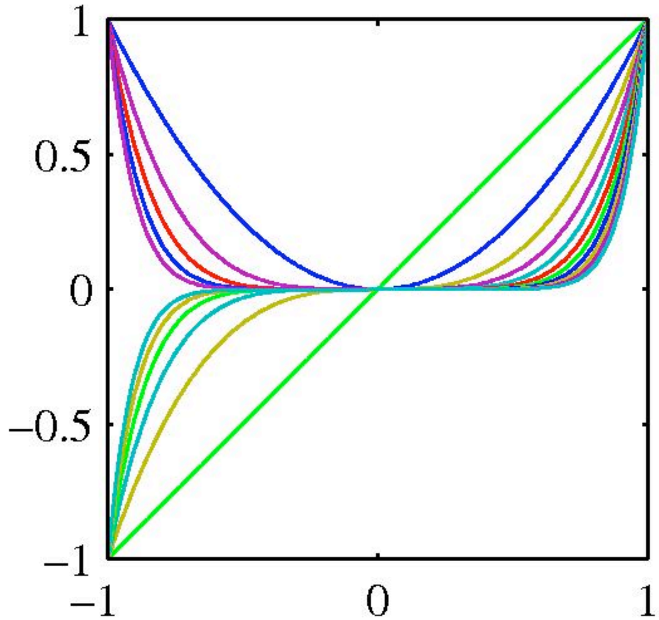
- Kernel trick produces efficient classification
- Dual formulation only assigns parameters to samples, not features



Linear Basis Function Models

Polynomial basis functions:

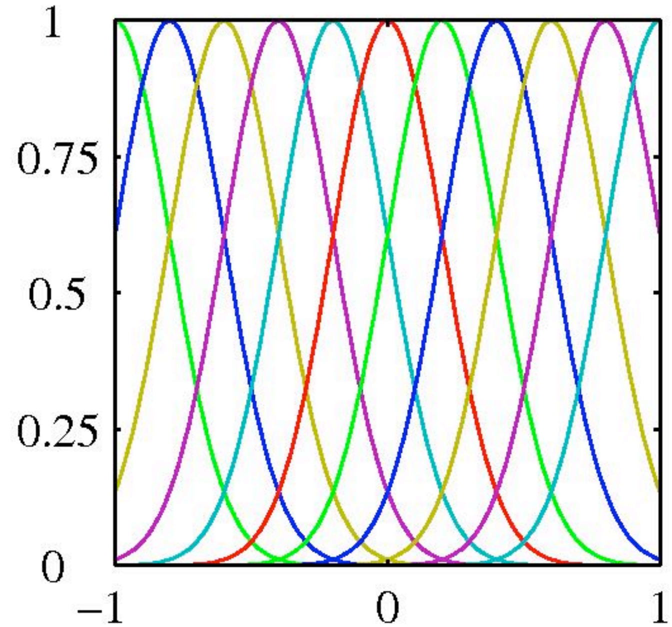
$$\phi_j(x) = x^j.$$



Basis functions are global: small changes in **x** affect all basis functions.

Gaussian basis functions:

$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right).$$

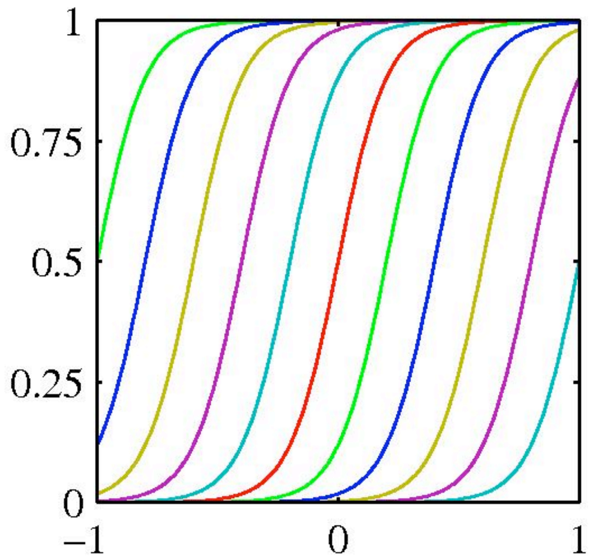


Basis functions are local: small changes in **x** only affect nearby basis functions.
μ_j and s control location and scale (width).

Linear Basis Function Models

Sigmoidal basis functions

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right), \text{ where } \sigma(a) = \frac{1}{1+\exp(-a)}.$$



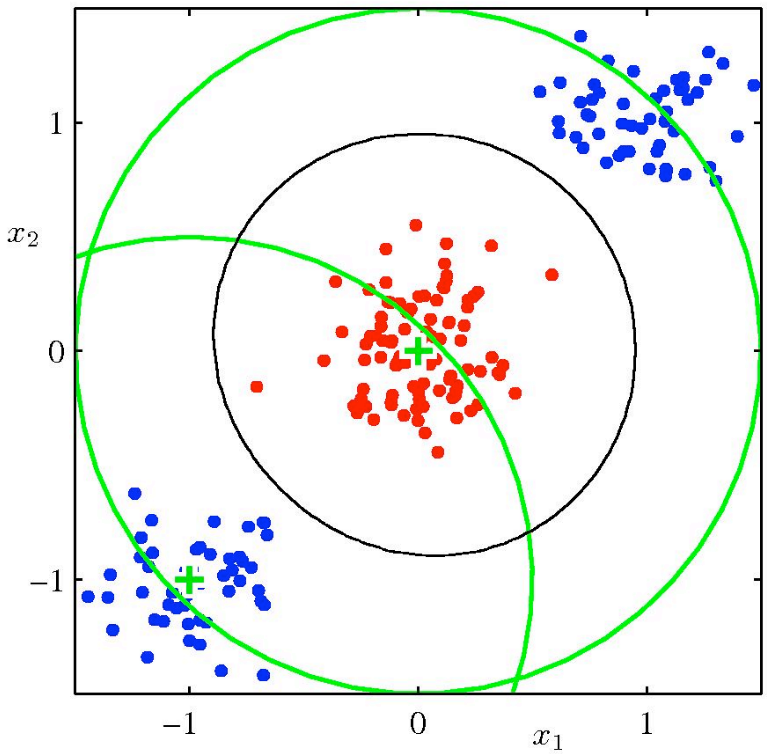
Basis functions are local: small changes in **x** only affect nearby basis functions. μ_j and s control location and scale (slope).

• Decision boundaries will be linear in the feature space $\phi,$ but would correspond to nonlinear boundaries in the original input space x.

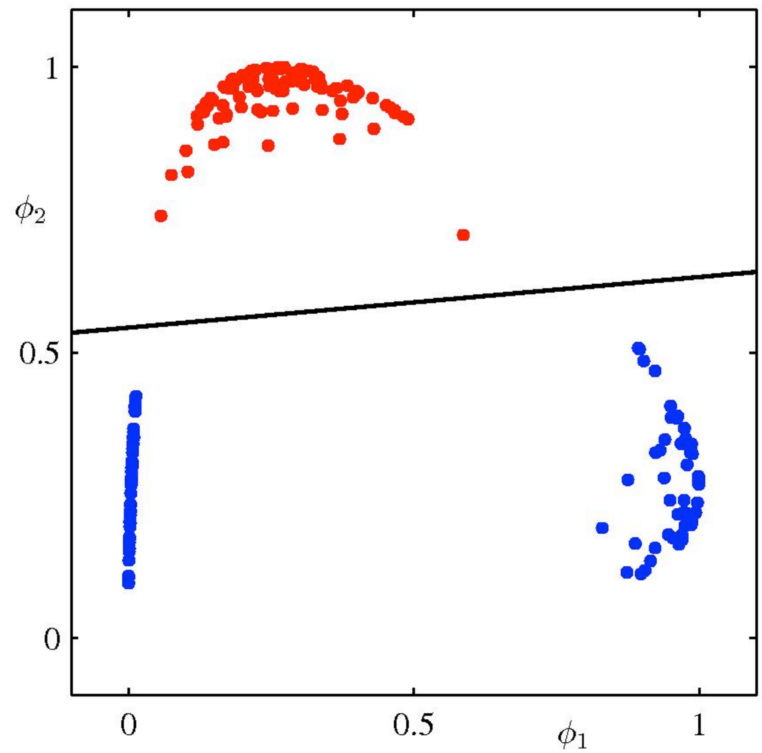
• Classes that are linearly separable in the feature space $\phi(\mathbf{x})$ need not be linearly separable in the original input space.

Linear Basis Function Models

Original input space



Corresponding feature space using
two Gaussian basis functions



- We define two Gaussian basis functions with centers shown by the green crosses, and with contours shown by the green circles.

- Linear decision boundary (right) is obtained by using logistic regression, and corresponds to the nonlinear decision boundary in the input space (left, black curve).

Classification with non-linear SVMs

Non-linear SVM using kernel function K():

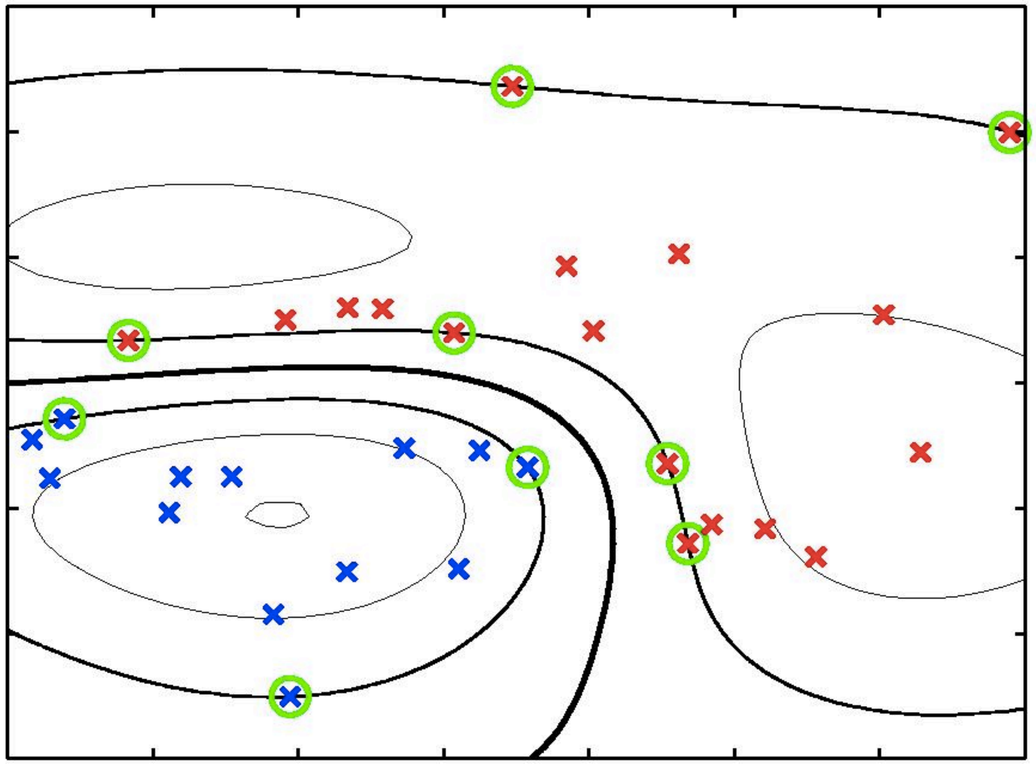
$$L_K = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Maximize L_K w.r.t. {α}, under constraints α≥0

Unlike linear SVM, cannot express w as linear combination of support vectors – now must retain the support vectors to classify new examples

Final decision function: $y = \text{sign}[b + \sum_{i=1}^{n} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i)]$

Classification with non-linear SVMs



- Synthetic data from two classes showing contours of constant y(x) obtained from an SVM having a Gaussian kernel function.
- Also shown are the decision boundary, the margin boundaries, and the support vectors.

Kernel Functions

Mercer's Theorem (1909): any reasonable kernel corresponds to some feature space

Reasonable means that the Gram matrix is positive definite

$$K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$$

Feature space can be very large, e.g., polynomial kernel $(1+\mathbf{x}_i + \mathbf{x}_j)^d$ corresponds to feature space exponential in d

Linear separators in these super high-dim spaces correspond to highly nonlinear decision boundaries in input space

Summary

Advantages:

- Kernels allow very flexible hypotheses
- Poly-time exact optimization methods rather than approximate methods
- Soft-margin extension permits mis-classified examples
- Variable-sized hypothesis space
- Excellent results (1.1% error rate on handwritten digits vs. LeNet's 0.9%)

Disadvantages:

- Must choose kernel parameters
- Very large problems computationally intractable
- Batch algorithm

More Summary

Software:

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

Key points:

- Difference between logistic regression and SVMs
- Maximum margin principle
- Target function for SVMs
- Slack variables for mis-classified points
- Kernel trick allows non-linear generalizations