
Survey of Classifiers

Raquel Romano

romano@hpcrd.lbl.gov

Machine Learning Reading Group

January 10, 2007



Generative Classifiers

- Model probability of a class given the data

$$P(k_i | \mathbf{x}) = \frac{P(\mathbf{x} | k_i)P(k_i)}{\sum_j P(\mathbf{x} | k_j)P(k_j)}$$

- Decision theory: estimate log likelihood ratio

$$L = \log \frac{P(k_1 | \mathbf{x})}{P(k_2 | \mathbf{x})}$$

- In general, too many variables to estimate!
- Make assumptions about probability densities and decision boundaries

Naïve Bayes

- Assume attributes/features are conditionally independent of each other, given class membership

$$P(\mathbf{x} | k_i) = \prod_j P(x_j | k_i)$$

- Decision rule simplifies to

$$\arg \max_{k_i} P(k_i) \prod_j P(x_j | k_i)$$

Pros

- Fast learning and classification, especially when feature space is high-dimensional
- Easy to compute densities and combine different variable types
- Explicit theoretical foundation
- Often performs well even when assumptions fail

Cons

- Conditional independence assumption is often wrong
- May perform poorly when assumptions fail

January 10, 2007



Linear Classifiers

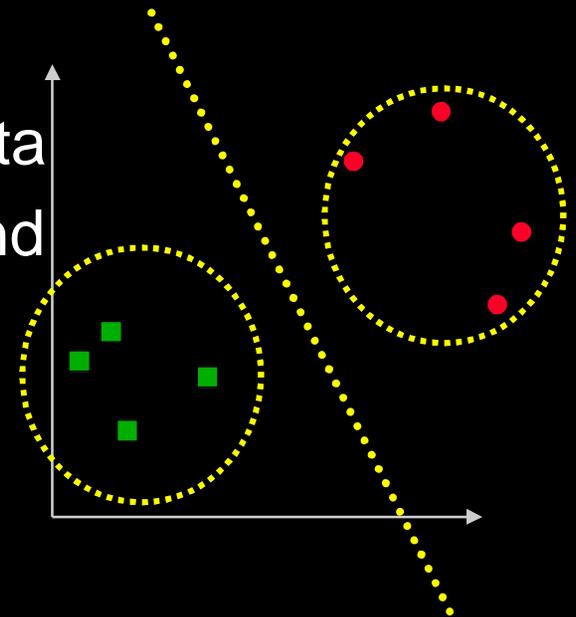
Linear Discriminant Analysis (LDA)

- Assume 2 classes have Gaussian densities with equal covariances
- Log likelihood ratio becomes linear in data
- Need to estimate class priors, means, and covariance

Fisher Linear Discriminant

- Formulate in subspace containing class centroids
- Decompose into series of 1D projections
- Maximize Rayleigh quotient:

$$\frac{\mathbf{a}^T \mathbf{B} \mathbf{a}}{\mathbf{a}^T \mathbf{W} \mathbf{a}}$$



Discriminative Classifiers

- Make assumptions about decision boundary, rather than class densities
- Optimize parameters describing decision boundary, e.g. hyperplane

January 10, 2007



Linear Discriminative Classifiers

Logistic regression

- Explicitly model log likelihood ratio as linear decision boundary

$$L = \log \frac{P(k_i | \mathbf{x})}{P(k_j | \mathbf{x})} = b_i + \mathbf{w}_i^T \mathbf{x}$$

- Class densities

$$P(k_0 | \mathbf{x}) = \frac{1}{1 + \sum_j \exp(b_j + \mathbf{w}_j^T \mathbf{x})}$$

$$P(k_i | \mathbf{x}) = \frac{\exp(b_i + \mathbf{w}_i^T \mathbf{x})}{1 + \sum_j \exp(b_j + \mathbf{w}_j^T \mathbf{x})}$$

- Guaranteed to find solution if linearly separable

Linear Discriminative Classifiers

Perceptron

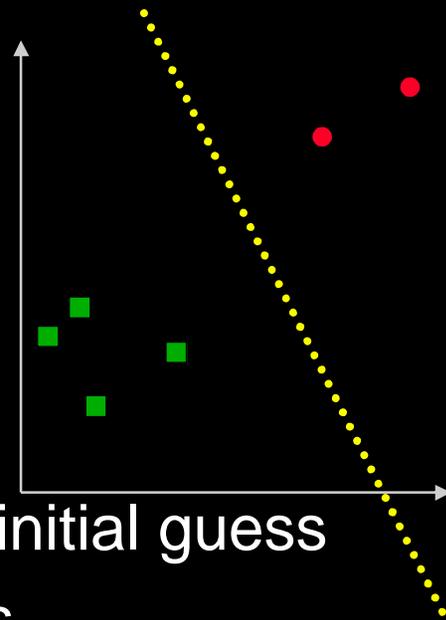
- Minimize distance of misclassified points to decision boundary

$$- \sum_i y_i (\mathbf{x}_i^T \mathbf{w} + b)$$

$y_i = 1, \quad \mathbf{x}_i^T \mathbf{w} + b > 0 \quad \text{for } i \in c_+$
 $y_i = -1, \quad \mathbf{x}_i^T \mathbf{w} + b < 0 \quad \text{for } i \in c_-$

- Stochastic gradient descent

$$\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix} \leftarrow \begin{pmatrix} \mathbf{w} \\ b \end{pmatrix} + \rho \begin{pmatrix} y_i \mathbf{x}_i \\ y_i \end{pmatrix}$$

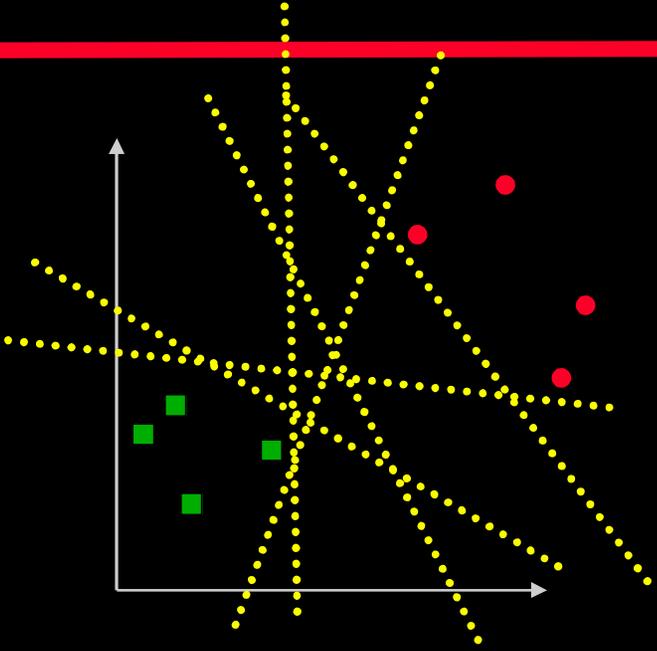


- Solution not unique; depends on initial guess
- If not separable, trapped in cycles

Linear Discriminative Classifiers

Linear SVM: Margin Maximization

- compute the **optimal** separating hyperplane between data points belonging to two classes



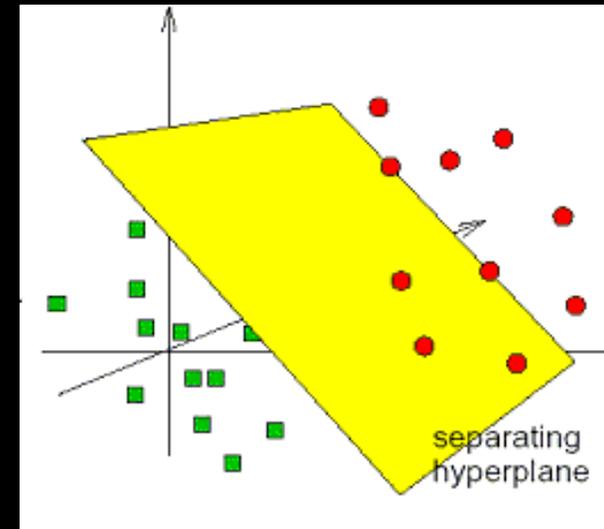
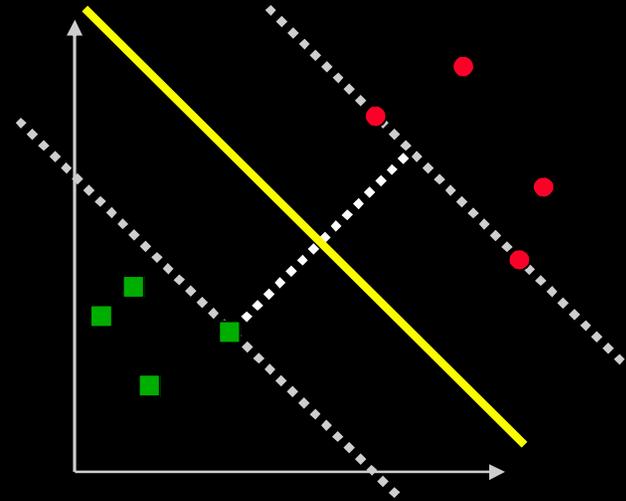
Linear

Linear SVM: Margin Maximization

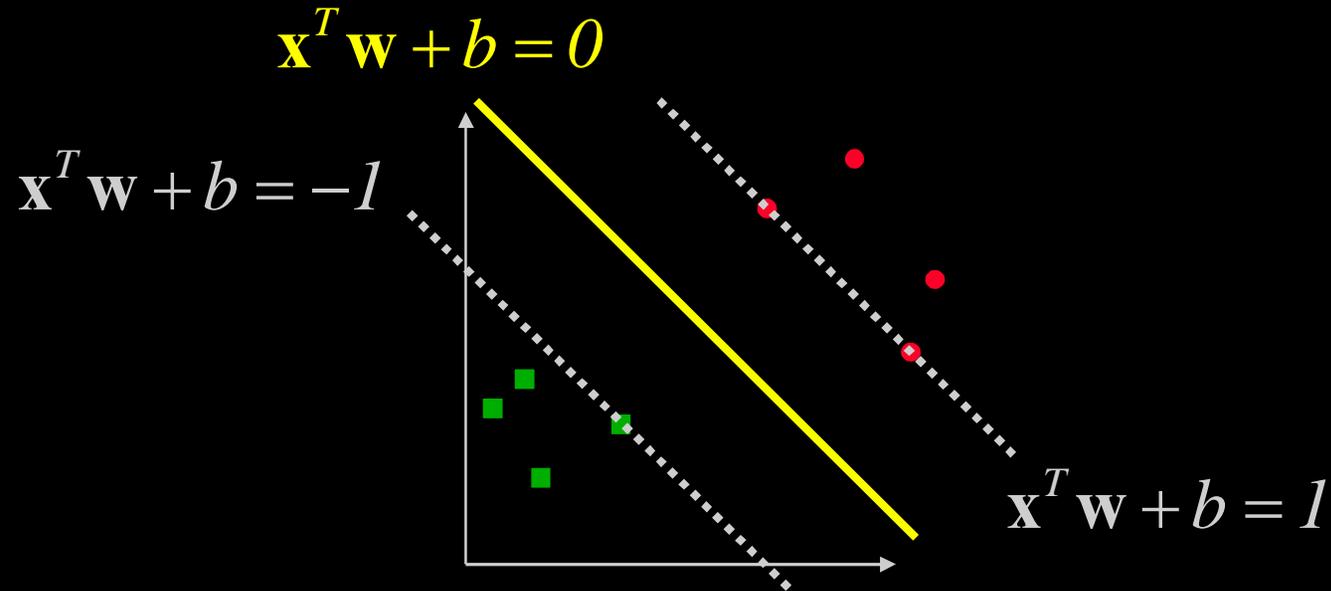
- compute the **optimal** separating hyperplane between data points belonging to two classes

Optimal Hyperplane

- maximize the distance from the decision surface to the nearest point in each class
- orthogonal to shortest line between convex hulls
- maximum margin separation



Margin Maximization



- Maximize distance between two parallel supporting planes
- Distance = margin = $\frac{2}{\|\mathbf{w}\|}$

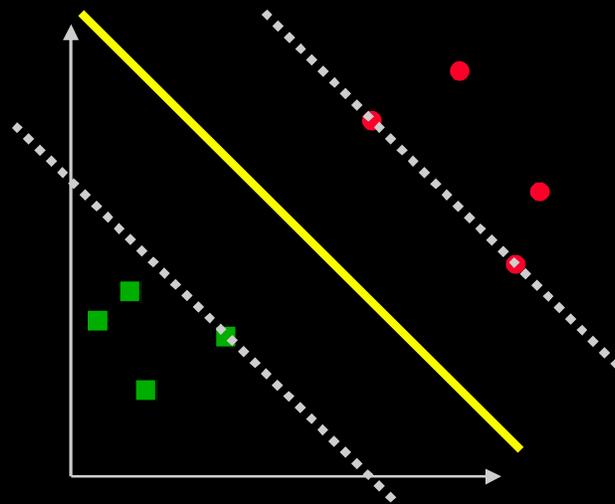
Constrained Optimization Problem

Objective Function

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to}$$
$$\mathbf{x}_i^T \mathbf{w} + b \geq 1 \quad \text{for } i \in \mathcal{C}_+$$
$$\mathbf{x}_i^T \mathbf{w} + b \leq -1 \quad \text{for } i \in \mathcal{C}_-$$

Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i \alpha_i (y_i (\mathbf{x}_i^T \mathbf{w} + b) - 1)$$



- $y_i = 1$ for $i \in \mathcal{C}_+$
- $y_i = -1$ for $i \in \mathcal{C}_-$

Lagrangian Formulation

Lagrangian

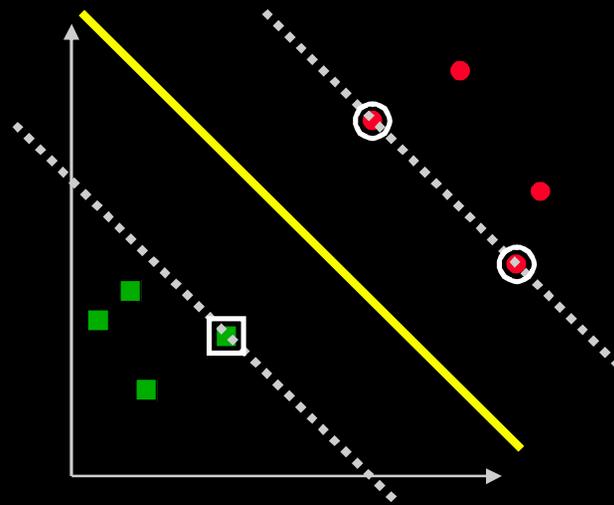
$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i \alpha_i (y_i (\mathbf{x}_i^T \mathbf{w} + b) - 1)$$

Differentiate

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0$$

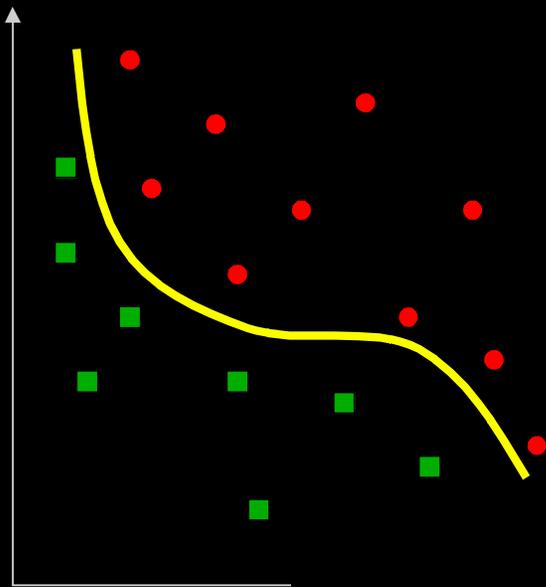
$$\Rightarrow \sum_i \alpha_i y_i = 0, \quad \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

Solution depends only on **support vectors** \mathbf{x}_i s.t. $\alpha_i > 0$



Nonlinear Decision Boundary

- Reality: classes may not be linearly separable



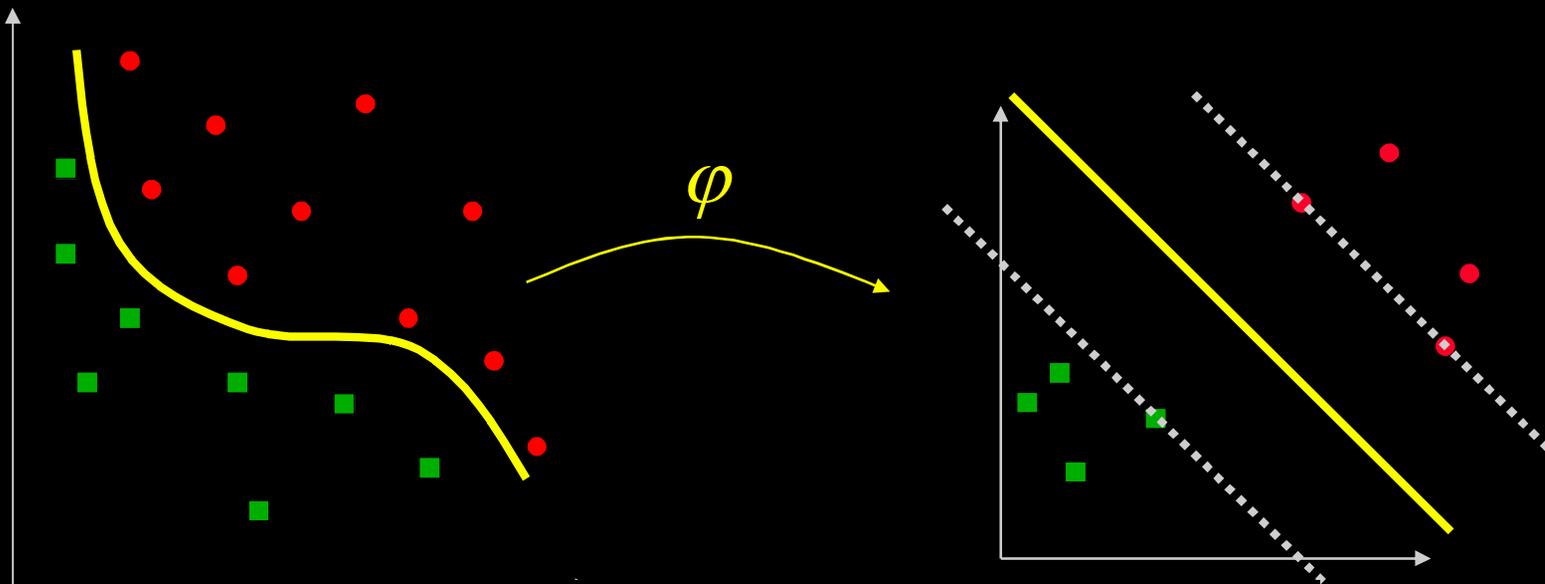
Chen, Lin, Schölkopf, *A Tutorial on ν -Support Vector Machines*, 2003.

Nonlinear Decision Boundary

- Reality: classes may not be linearly separable
- Map points to a higher-dimensional feature space
e.g. products up to degree d

$$\mathbf{x}^T = [x_1, x_2]$$

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



Chen, Lin, Schölkopf, *A Tutorial on ν -Support Vector Machines*, 2003.

Nonlinear Decision Boundary

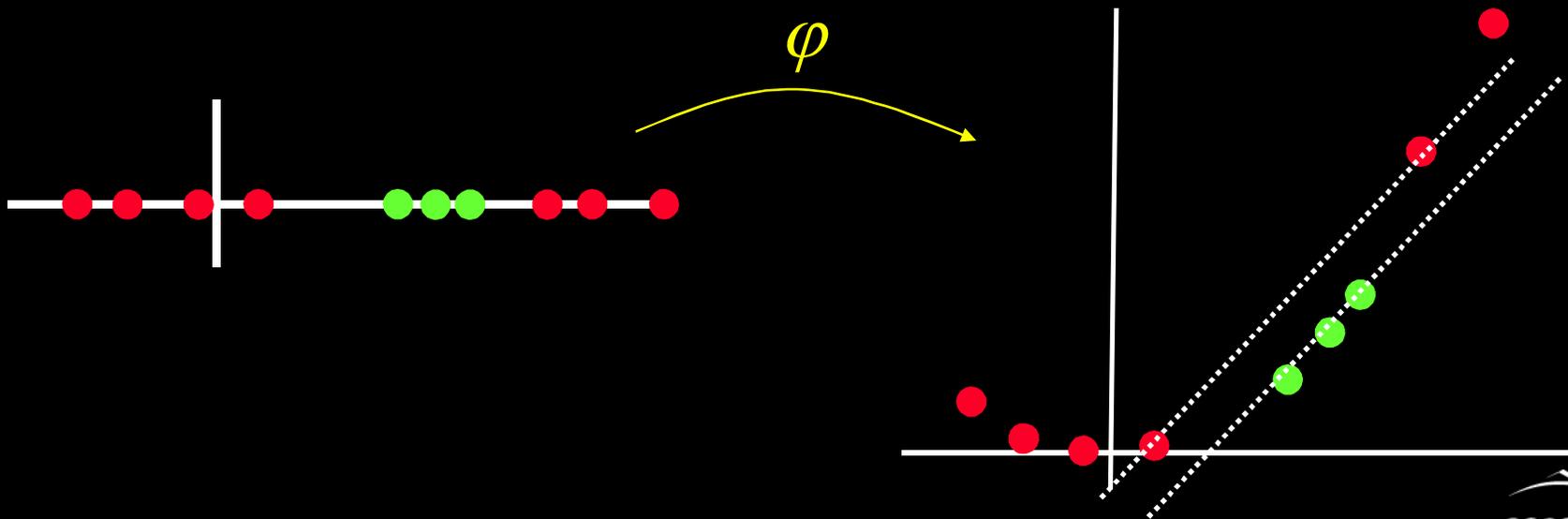
- Map points to a higher-dimensional feature space
- Generalized inner product is called a *kernel*

$$\mathbf{x}^T = [x_1, x_2]$$

$$\mathbf{x}^T \mathbf{w} = x_1 w_1 + x_2 w_2$$

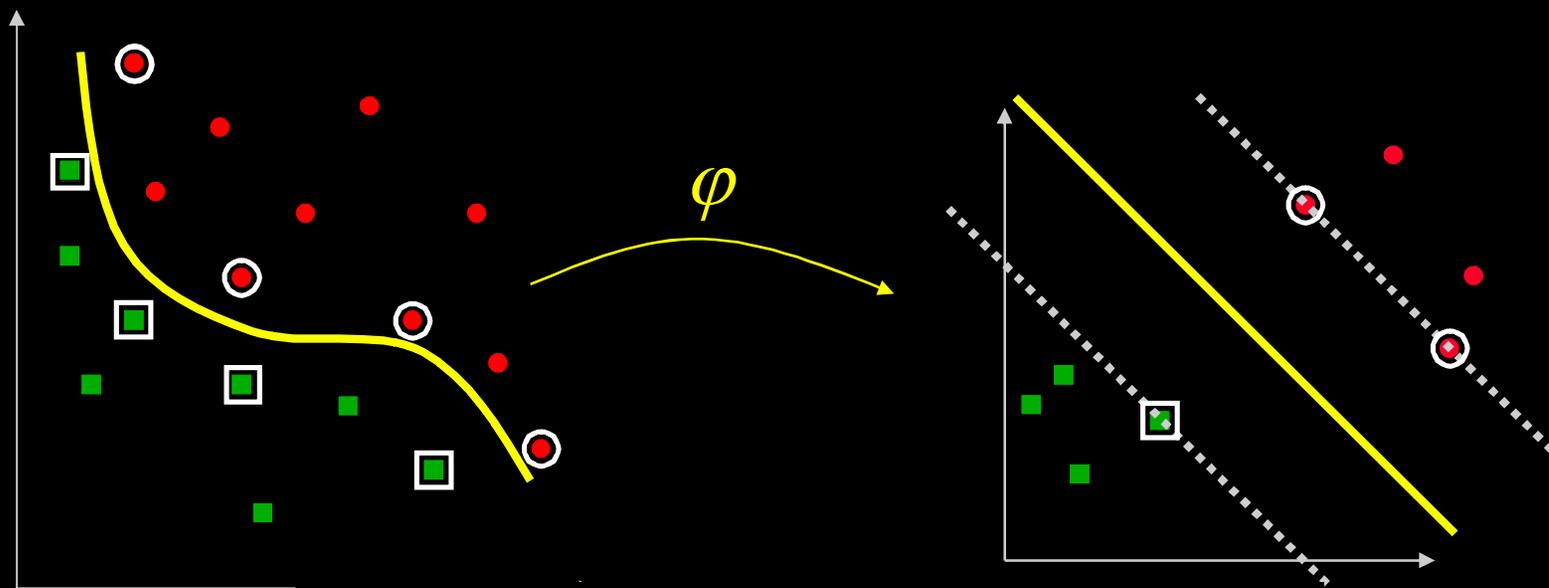
$$\varphi(\mathbf{x})^T = [x_1^2, x_1 x_2, x_2^2]$$

$$\text{kernel } k(\mathbf{x}, \mathbf{w}) = \varphi(\mathbf{x})^T \varphi(\mathbf{w})$$

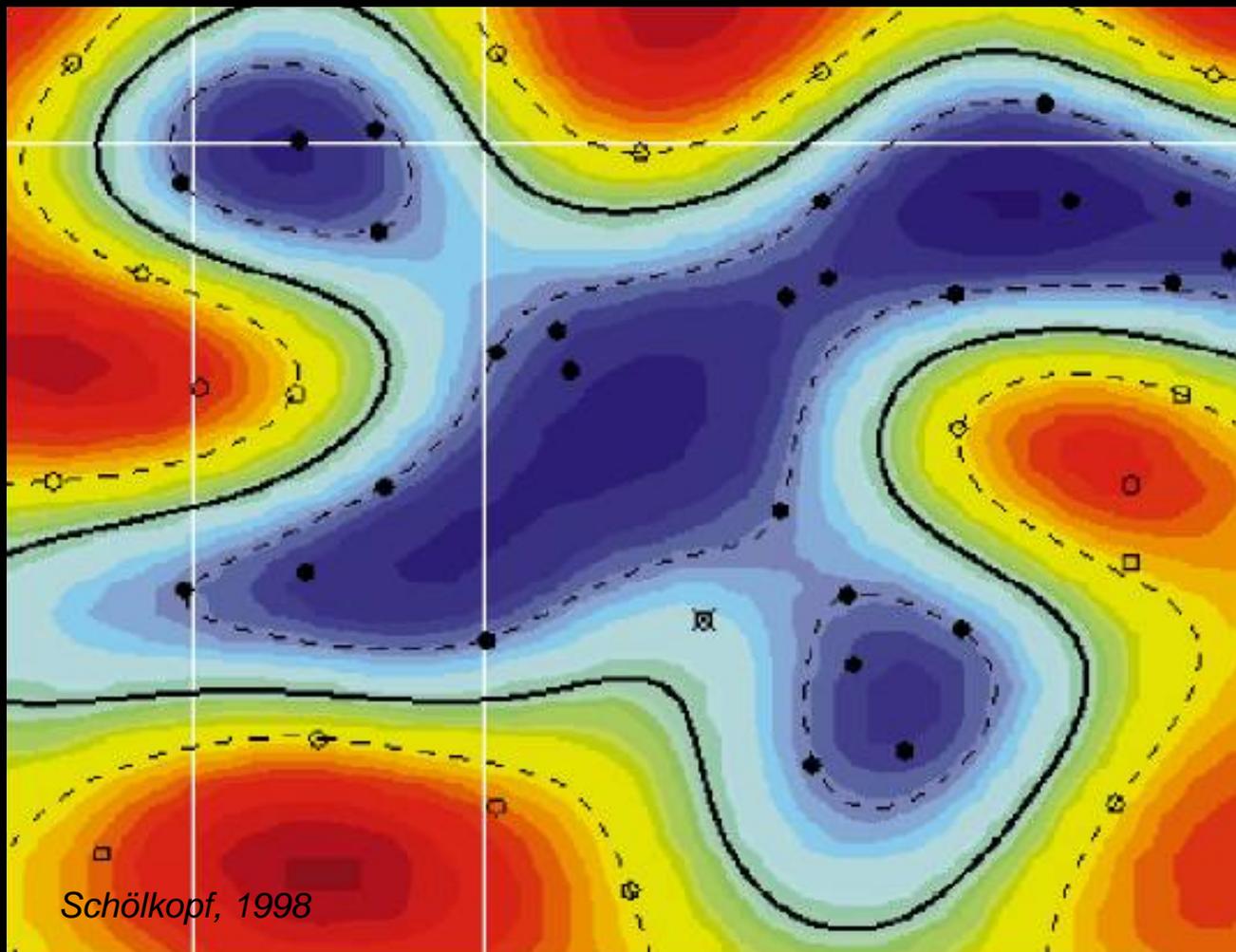


Kernels

- Can explicitly define kernel $\varphi(\mathbf{x})^T \varphi(\mathbf{w}) = k(\mathbf{x}, \mathbf{w})$ to induce implicit mapping φ
- Gaussian radial basis function $k(\mathbf{x}, \mathbf{w}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{w}\|^2}{2\sigma^2}\right)$
- Decision boundary is a linear combination of support vectors, optimally chosen from training set



Example: 2D RBF



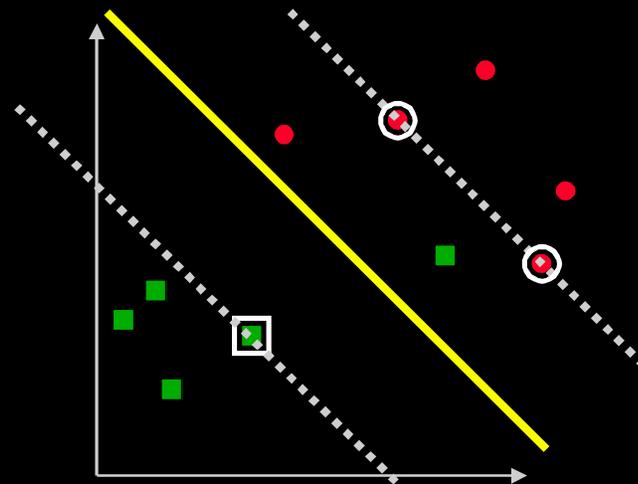
January 10, 2007

Handling Real Data

- No separating hyperplane, even after mapping!
- **Soft margin classifiers**
 - Slack variables allowing points to lie inside margin:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \quad \text{subject to} \quad y_i(\mathbf{x}_i^T \mathbf{w} + b) \geq 1 - \xi_i$$

- Or: penalty terms for number of examples that are support vectors, number of examples on wrong side of hyperplane



January 10, 2007

Support Vector Machines

Pros

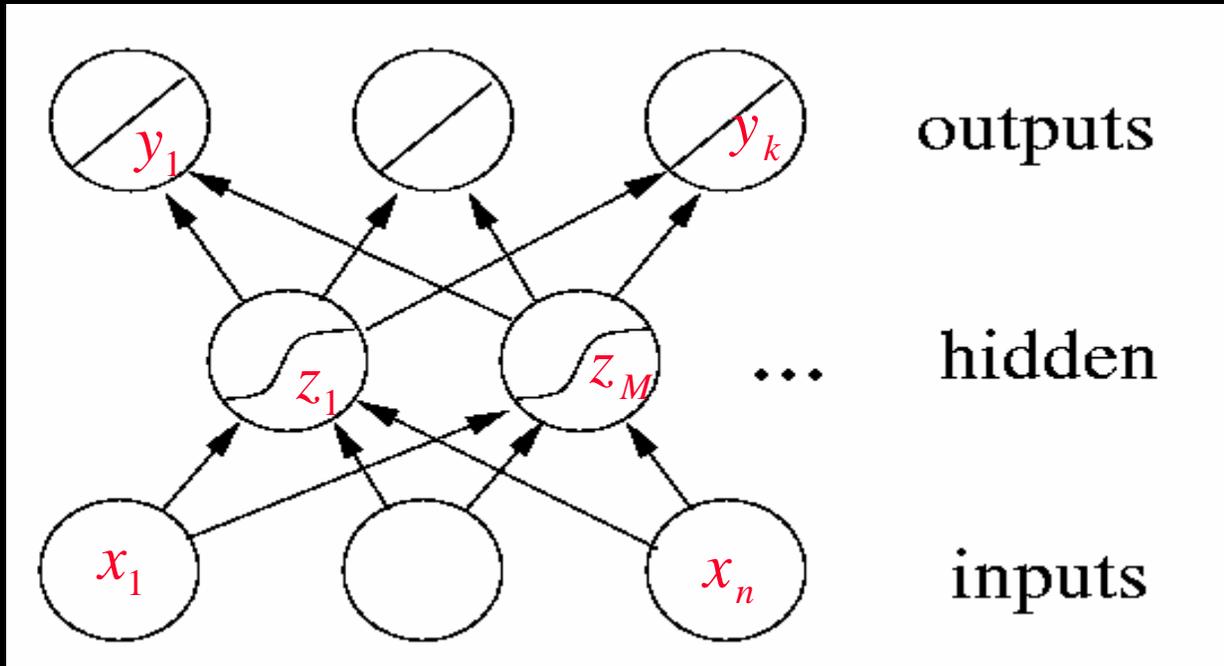
- Arbitrarily complicated decision boundaries
- Convex optimization so unique optimum
- Handles overlapping classes

Cons

- Parameters must be tuned by cross-validation
- Kernel is chosen empirically
- Solution may be difficult to interpret

Neural Networks

Multi-Layer Perceptron single hidden layer back-propagation network



$$f_k(\mathbf{x}) = g_k(\beta_{0k} + \beta_k^T \mathbf{z})$$

$$z_m = \sigma(\alpha_{0m} + \alpha_m^T \mathbf{x})$$

nonlinearity

- hidden units are functions of linear combinations of inputs
- parameters learned from data by minimizing error function

$$R(\theta) = \sum_k \sum_i (y_{ik} - f_k(\mathbf{x}_i))^2$$

- gradient descent (typical back-propagation) or other optimization algorithms

Neural Networks

Pros

- Can model any continuous real function arbitrarily well
- Easily parallelized due to local nature of weight updates
- Handles many features and many classes

Cons

- Sensitive to initial parameter values
- Generally over-parameterized so susceptible to overfitting
- Architecture design largely empirical; trial and error
- Nonconvex error function with many local minima
- Training can be slow