Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- Discriminative learning II
  - Kernel SVM

  If time:
  - Multi-class classification
    - Using binary classifiers
    - Using multi-class SVM
  - Ranking
    - SVM Rank

with much credit to T. Jaakkola

Kernel perceptron review

- Suppose the training set is linearly separable through origin given a particular feature mapping, i.e.,

$$y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) > 0, \; i = 1, \ldots, n$$

for some $\underline{\theta}$

- The perceptron algorithm, applied repeatedly over the training set, will find a solution of the form

$$\underline{\theta} = \sum_{i=1}^{n} \alpha_i \, \underline{y_i \underline{\phi}(\underline{x}_i)}, \quad \alpha_i \in \{0, 1, \ldots\}$$

the number of mistakes made on
the ith training example until convergence

- We can rewrite the algorithm solely in terms of these "mistake counts" α_i

Kernel perceptron

- We don't need the parameters nor the feature vectors explicitly
- All we need for predictions as well as updates is the value of the discriminant function

$$\underline{\theta} \cdot \underline{\phi}(\underline{x}) = \sum_{i=1}^{n} \alpha_i y_i \underline{[\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x})]} = \sum_{i=1}^{n} \alpha_i y_i \underbrace{K(\underline{x}_i, \underline{x})}_{\text{kernel}}$$

$$\text{Initialize: } \alpha_i = 0,\ i = 1, \ldots, n$$

$$\text{Repeat for } t = 1, \ldots, n$$

$$\text{if } y_t \underline{\left(\textstyle\sum_{i=1}^{n} \alpha_i y_i K(\underline{x}_i, \underline{x}_t) \right)} \leq 0 \ (\text{mistake})$$

$$\alpha_t \leftarrow \alpha_t + 1$$

value of the discriminant function prior to the update

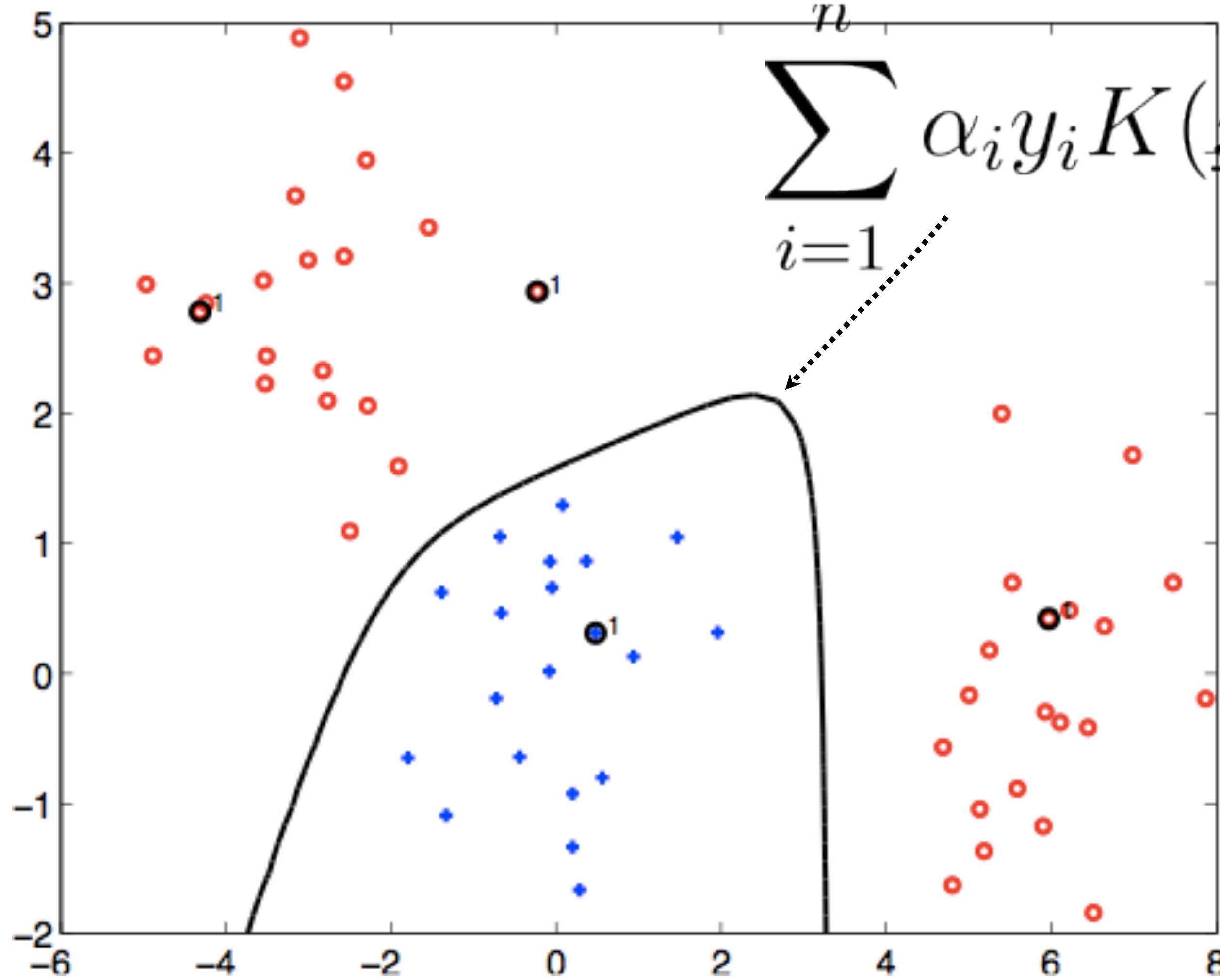
Kernel perceptron: example

- With a radial basis kernel

$$f(\underline{x};\alpha) = \text{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i K(\underline{x}_i, \underline{x}) \Big)$$

Decision surface:

$$\sum_{i=1}^{n} \alpha_i y_i K(\underline{x}_i, \underline{x}) = 0$$

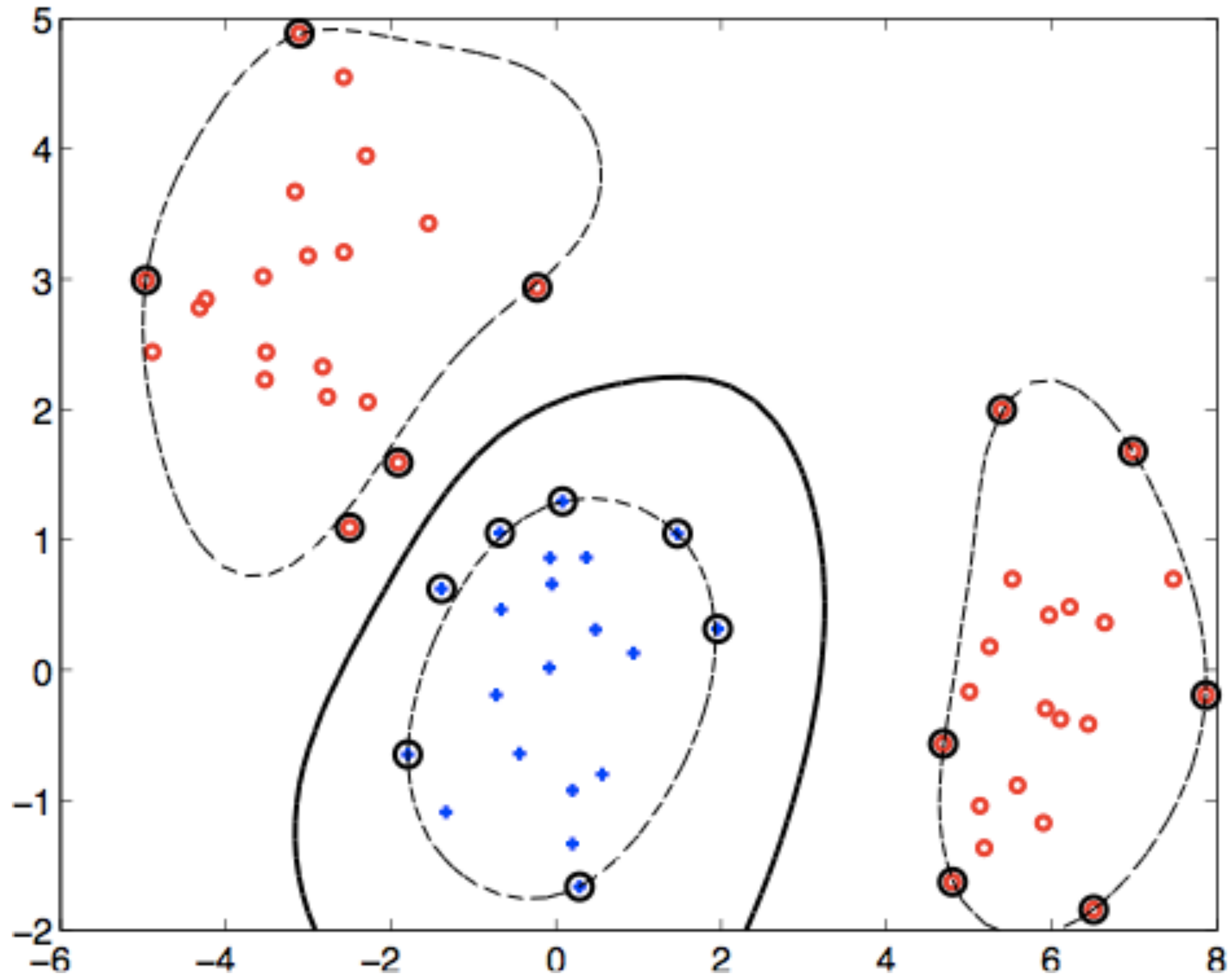


Only 4 mistakes!

Kernel SVM

- Kernel SVM: implicitly find the max-margin linear separator in the feature space, e.g., corresponding to the radial basis kernel

$$f(\underline{x};\alpha) = \mathrm{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i K(\underline{x}_i, \underline{x}) + \theta_0 \Big)$$



Kernels

- By writing the algorithm in a "kernel" form, we can try to work with the kernel (inner product) directly rather than expanding the high dimensional feature vectors

$$
\begin{aligned}
K(\underline{x}, \underline{x}') &= \underline{\phi}(\underline{x}) \cdot \underline{\phi}(\underline{x}') \\
&= \begin{bmatrix} \\ ? \\ \\ \end{bmatrix} \cdot \begin{bmatrix} \\ ? \\ \\ \end{bmatrix} \\
&= \exp(-\|\underline{x} - \underline{x}'\|^2) \quad \text{(e.g.)}
\end{aligned}
$$

- All we need to ensure is that the kernel is "valid", i.e., there exists some underlying feature representation, ϕ

Valid kernels

- A kernel function is valid (is a kernel) if there exists some feature mapping such that

$$K(\underline{x}, \underline{x}') \quad = \quad \underline{\phi}(\underline{x}) \cdot \underline{\phi}(\underline{x}')$$

- We can verify this, e.g., via the composition rules
- Equivalently, a kernel is valid if it is symmetric and for all training sets, the Gram matrix

$$\begin{bmatrix} K(\underline{x}_1, \underline{x}_1) & \cdots & K(\underline{x}_1, \underline{x}_n) \\ \cdots & \cdots & \cdots \\ K(\underline{x}_n, \underline{x}_1) & \cdots & K(\underline{x}_n, \underline{x}_n) \end{bmatrix}$$

is positive semi-definite

(Kernel) SVM: Primal problem

- Consider a simple max-margin classifier through origin

$$\text{minimize} \quad \frac{1}{2}\|\underline{\theta}\|^2 \quad \text{subject to}$$
$$y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) \geq 1, \quad i = 1, \ldots, n$$

- The solution has the same form as in the perceptron case!

$$\underline{\theta}(\alpha) = \sum_{i=1}^{n} \alpha_i \, y_i \underline{\phi}(\underline{x}_i), \quad \alpha_i \geq 0$$

non-negative Lagrange multipliers used to enforce the classification constraints

- As before, we focus on estimating α_i which are now non-negative real numbers (Lagrange multipliers)

Primal SVM

- Consider a simple max-margin classifier through origin

$$\text{minimize} \;\; \frac{1}{2}\,\|\underline{\theta}\|^2 \;\; \text{subject to}$$

$$y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) \geq 1, \;\; i = 1, \ldots, n$$

- To solve this, we can introduce Lagrange multipliers for the classification constraints and minimize the resulting Lagrangian with respect to the parameters $\underline{\theta}$

$$L(\underline{\theta}, \alpha) = \frac{1}{2}\,\|\underline{\theta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i) - 1 \right]$$

$$\alpha_i \geq 0, \;\; i = 1, \ldots, n$$

Primal SVM

- Consider a simple max-margin classifier through origin

$$\text{minimize} \quad \frac{1}{2}\|\underline{\theta}\|^2 \quad \text{subject to}$$
$$y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) \geq 1, \quad i = 1, \ldots, n$$

- To solve this, we can introduce Lagrange multipliers for the classification constraints and minimize the resulting Lagrangian with respect to the parameters $\underline{\theta}$

non-negative when constraint is met

$$L(\underline{\theta}, \alpha) = \frac{1}{2}\|\underline{\theta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i) - 1 \right]$$

$$\alpha_i \geq 0, \quad i = 1, \ldots, n$$

positive values enforce classification constraints

Understanding the Lagrangian

$$L(\underline{\theta}, \alpha) = \frac{1}{2} \|\underline{\theta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i) - 1 \right]$$

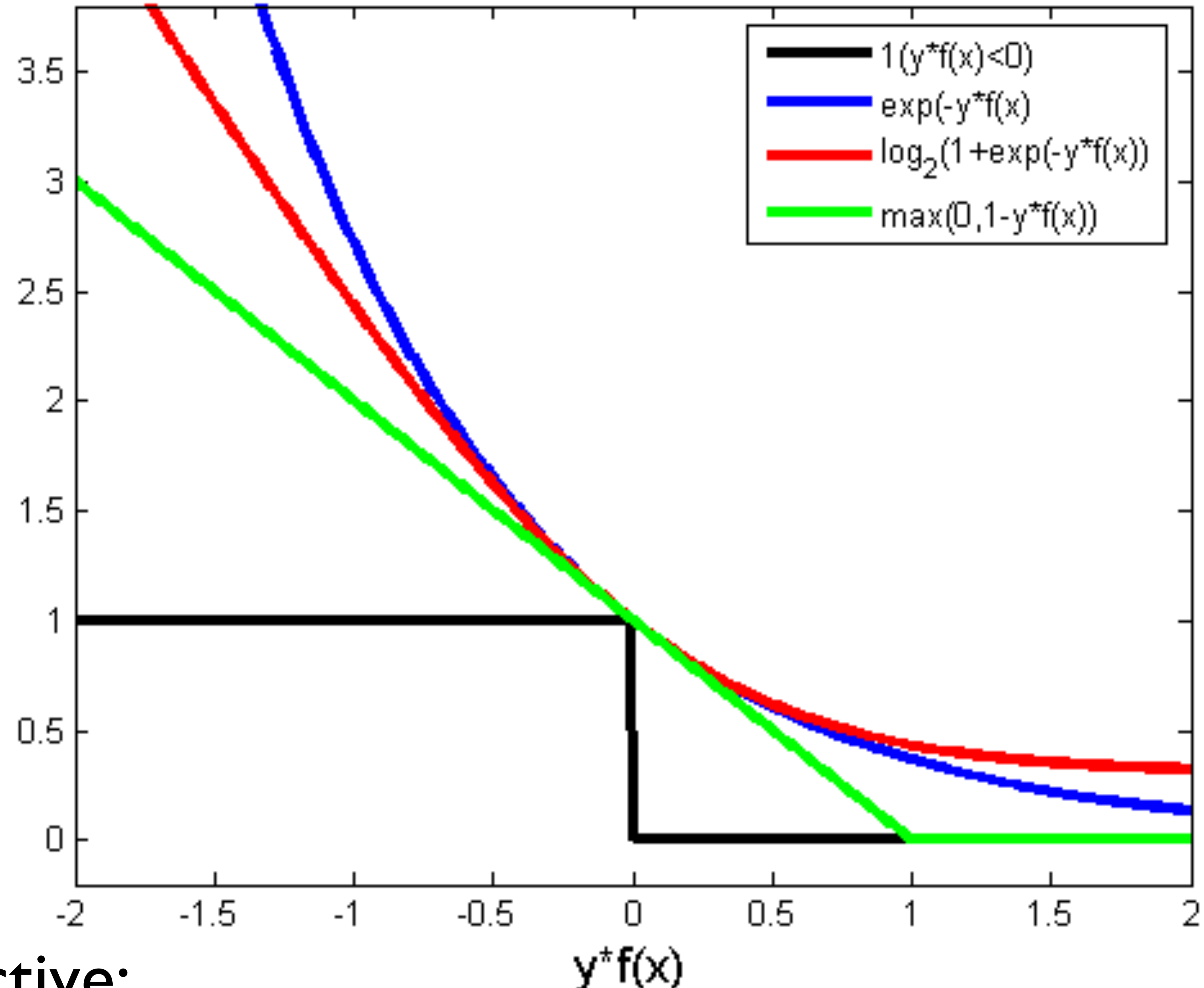
$$\alpha_i \geq 0, \quad i = 1, \ldots, n$$

- We can recover the primal problem by maximizing the Lagrangian with respect to the Lagrange multipliers α_i

$$\max_{\alpha \geq 0} L(\underline{\theta}, \alpha) = \begin{cases} \frac{1}{2} \|\underline{\theta}\|^2, & \text{if } \; y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) \geq 1, \; i = 1, \ldots, n \\ \infty, & \text{otherwise} \end{cases}$$

Loss functions

- SVM optimizes the Hinge Loss (shown in green)



Lagrangian objective:

$$L(\underline{\theta}, \alpha) = \frac{1}{2} \|\underline{\theta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i) - 1 \right]$$

Primal - Dual

$$\text{minimize } \frac{1}{2}||\underline{\theta}||^2 \text{ subject to}$$

$$y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) \geq 1, \quad i = 1, \ldots, n$$

$$\min_{\underline{\theta}} \overbrace{\left[\max_{\alpha \geq 0} L(\underline{\theta}, \alpha)\right]}^{\text{primal}(\underline{\theta})}$$

- expressed in terms of $\underline{\theta}$
- explicit feature vectors $\underline{\phi}(\underline{x})$

?

Primal - Dual

minimize $\frac{1}{2}\,||\underline{\theta}||^2$ subject to
$y_i(\underline{\theta}\cdot\phi(\underline{x}_i)) \geq 1,\quad i = 1,\ldots,n$

?

$$\min_{\underline{\theta}} \overbrace{\Big[\max_{\alpha\geq 0} L(\underline{\theta},\alpha)\Big]}^{\text{primal}(\underline{\theta})} = \underbrace{\max_{\alpha\geq 0}}_{\text{step 2}} \underbrace{\overbrace{\Big[\min_{\underline{\theta}} L(\underline{\theta},\alpha)\Big]}^{\text{dual}(\alpha)}}_{\text{step 1}}$$

Slater conditions

- expressed in terms of $\underline{\theta}$
- explicit feature vectors $\phi(\underline{x})$

- expressed in terms of α
- kernels $K(\underline{x},\underline{x}')$

Dual (step 1)

$$L(\underline{\theta}, \alpha) = \frac{1}{2}\left\|\underline{\theta}\right\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) - 1\right]$$

$$\frac{\partial}{\partial \underline{\theta}} L(\underline{\theta}, \alpha) \qquad = 0$$

Lagrangian Dual (step 1)

$$L(\underline{\theta}, \alpha) = \frac{1}{2} \|\underline{\theta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) - 1 \right]$$

$$\frac{\partial}{\partial \underline{\theta}} L(\underline{\theta}, \alpha) = \underline{\theta} - \qquad\qquad\qquad = 0$$

Lagrangian Dual (step 1)

$$L(\underline{\theta}, \alpha) = \frac{1}{2} \|\underline{\theta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i (\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) - 1 \right]$$

$$\frac{\partial}{\partial \underline{\theta}} L(\underline{\theta}, \alpha) = \underline{\theta} - \sum_{i=1}^{n} \alpha_i \, y_i \underline{\phi}(\underline{x}_i) = 0$$

Lagrangian Dual (step 1)

$$L(\underline{\theta}, \alpha) = \frac{1}{2} \|\underline{\theta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i (\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) - 1 \right]$$

$$\frac{\partial}{\partial \underline{\theta}} L(\underline{\theta}, \alpha) = \underline{\theta} - \sum_{i=1}^{n} \alpha_i \, y_i \underline{\phi}(\underline{x}_i) = 0$$

$$\Rightarrow \underline{\theta} = \sum_{i=1}^{n} \alpha_i \, y_i \underline{\phi}(\underline{x}_i) = \underline{\theta}(\alpha)$$

(unique solution as a function of α)

Lagrangian Dual (step 1)

$$L(\underline{\theta}, \alpha) = \frac{1}{2}\|\underline{\theta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i)) - 1 \right]$$

$$\frac{\partial}{\partial \underline{\theta}} L(\underline{\theta}, \alpha) = \underline{\theta} - \sum_{i=1}^{n} \alpha_i \, y_i \underline{\phi}(\underline{x}_i) = 0$$

$$\Rightarrow \underline{\theta} = \sum_{i=1}^{n} \alpha_i \, y_i \underline{\phi}(\underline{x}_i) = \underline{\theta}(\alpha)$$

(unique solution as a function of α)

- The dual problem is obtained by substituting this solution back into the Lagrangian and recalling that the Lagrange multipliers are non-negative

$$\mathrm{dual}(\alpha) = \min_{\underline{\theta}} L(\underline{\theta}, \alpha) = L(\underline{\theta}(\alpha), \alpha) \quad \alpha_i \geq 0$$

Dual SVM (step 2)

- Recall the Lagrangian:

$$L(\underline{\theta}, \alpha) = \frac{1}{2}\|\underline{\theta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\underline{\theta} \cdot \underline{\phi}(\underline{x}_i) - 1\right]$$

$$\alpha_i \geq 0, \quad i = 1, \ldots, n$$

- Plug in: $$\underline{\theta} = \sum_{i=1}^{n} \alpha_i \, y_i \underline{\phi}(\underline{x}_i)$$

$$L = \frac{1}{2}\|\sum_{i=1}^{n} \alpha_i y_i \underline{\phi}(\underline{x}_i)\|^2 - \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x}_j)) + \sum_{i=1}^{n} \alpha_i$$

$$= -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x}_j)) + \sum_{i=1}^{n} \alpha_i$$

Dual SVM (step 2)

$$\underline{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j [\underbrace{\phi(\underline{x}_i) \cdot \phi(\underline{x}_j)}_{\text{kernel}}]$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n$$

- This is again a quadratic programming problem but with simpler "box" constraints

Dual SVM (step 2)

$$\underline{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j [\underbrace{\phi(\underline{x}_i) \cdot \phi(\underline{x}_j)}_{\text{kernel}}]$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n$$

- This is again a quadratic programming problem but with simpler "box" constraints

the solution α^* is not necessarily unique

Dual SVM (step 2)

$$\underline{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j [\underbrace{\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x}_j)}_{\text{kernel}}]$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n$$

- This is again a quadratic programming problem but with simpler "box" constraints

the solution α^* is not necessarily unique

$$\underline{\theta}(\alpha^*) = \sum_{i=1}^{n} \alpha_i^* y_i \underline{\phi}(\underline{x}_i) \quad \text{is unique} \quad (= \underline{\theta}^*)$$

Dual SVM (step 2)

$$\underline{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j [\underbrace{\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x}_j)}_{\text{kernel}}]$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n$$

- This is again a quadratic programming problem but with simpler "box" constraints

the solution α^* is not necessarily unique

$$\underline{\theta}(\alpha^*) = \sum_{i=1}^{n} \alpha_i^* y_i \underline{\phi}(\underline{x}_i) \quad \text{is unique} \quad (= \underline{\theta}^*)$$

$$\text{if } \alpha_i^* > 0 \quad \Rightarrow \quad y_i(\underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x}_i)) = 1 \qquad \text{(support vector)}$$

Dual SVM (step 2)

$$\underline{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j [\underbrace{\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x}_j)}_{\text{kernel}}]$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n$$

- This is again a quadratic programming problem but with simpler "box" constraints

the solution α^* is not necessarily unique

$$\underline{\theta}(\alpha^*) = \sum_{i=1}^{n} \alpha_i^* y_i \underline{\phi}(\underline{x}_i) \quad \text{is unique} \quad (= \underline{\theta}^*)$$

$$\text{if } \alpha_i^* > 0 \quad \Rightarrow \quad y_i(\underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x}_i)) = 1 \qquad \text{(support vector)}$$

$$\text{if } \alpha_i^* = 0 \quad \Rightarrow \quad y_i(\underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x}_i)) \geq 1$$

Dual SVM

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j [\underbrace{\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x}_j)}_{\text{kernel}}]$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n$$

- Once we solve for α_i^*, we predict labels according to:

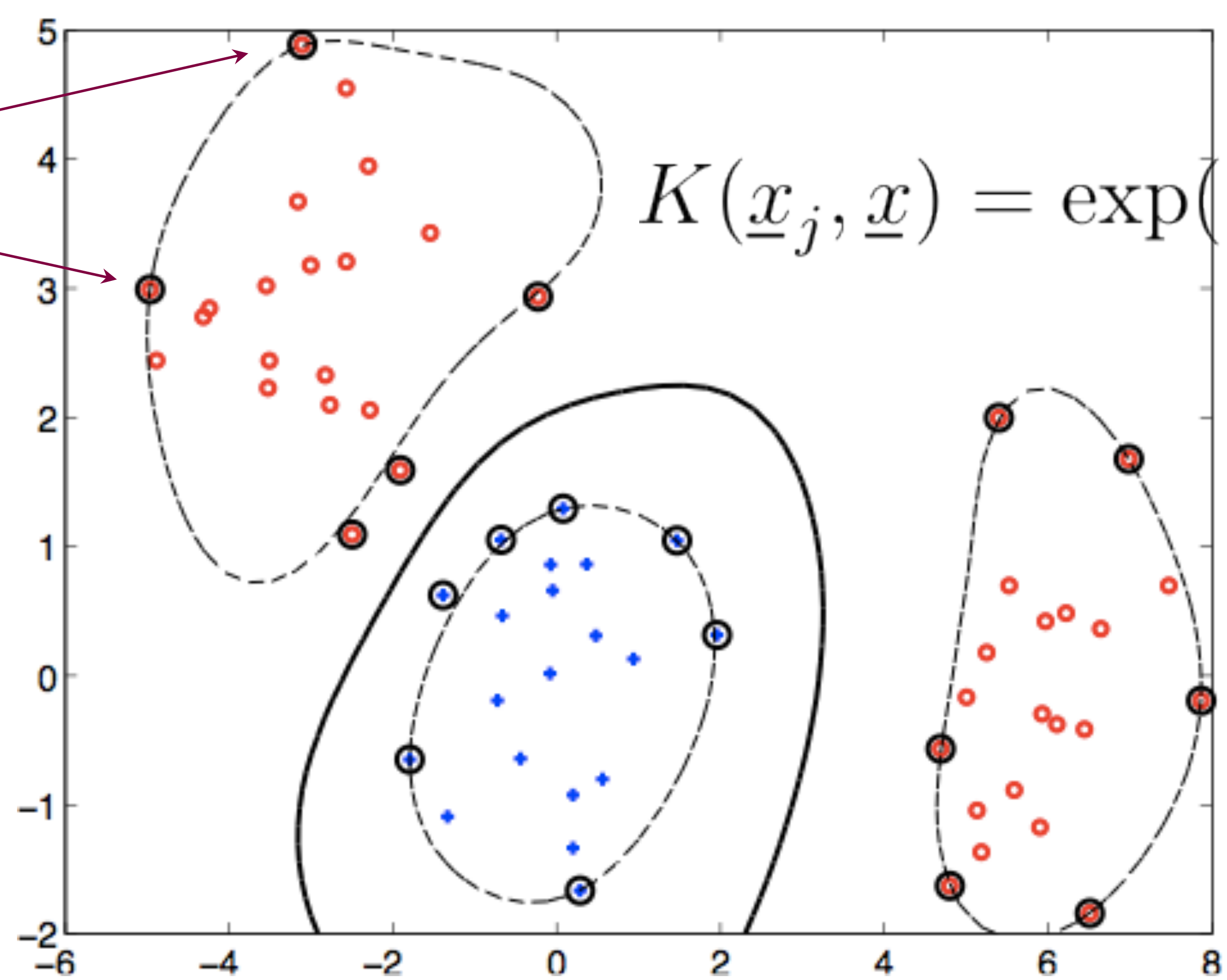
$$\begin{aligned} f(\underline{x}; \alpha^*) &= \operatorname{sign}\Big(\underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x})\Big) \\ &= \operatorname{sign}\Big(\sum_{i=1}^{n} \alpha_i^* y_i [\underbrace{\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x})}_{\text{kernel}}]\Big) \end{aligned}$$

Kernel SVM

- Solving the SVM dual implicitly finds the max-margin linear separator in the feature space

$$f(\underline{x};\alpha) = \mathrm{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i K(\underline{x}_i, \underline{x})\Big)$$

support vectors

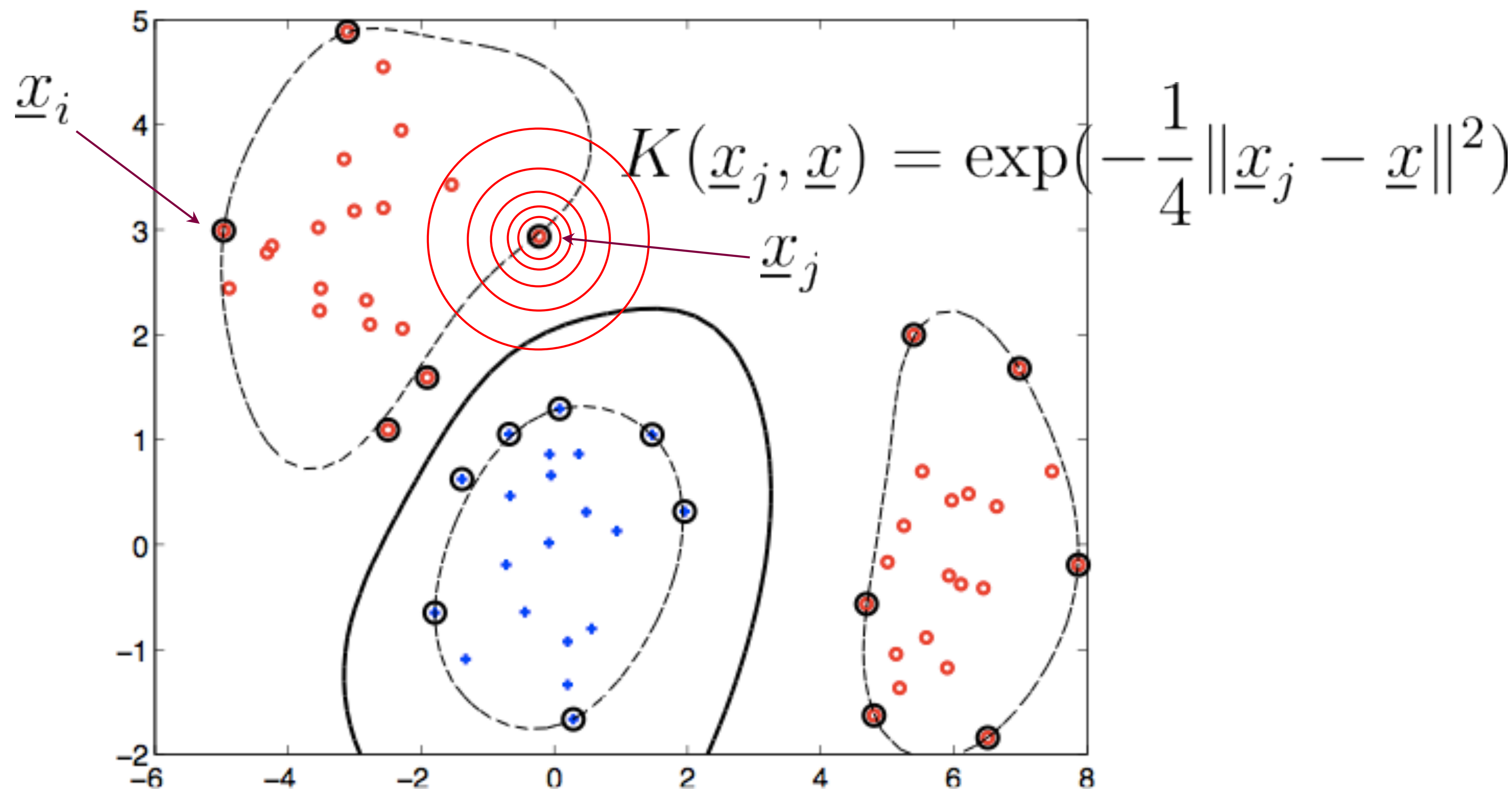


$$K(\underline{x}_j, \underline{x}) = \exp\big(-\frac{1}{4}\|\underline{x}_j - \underline{x}\|^2\big)$$

RBF kernel, support vectors

- Assume no offset, no slack. A point is not a support vector if its margin constraint is satisfied (otherwise it has to be a SV)

$$y_i \sum_{j \neq i} \alpha_j y_j K(\underline{x}_j, \underline{x}_i) \geq 1 \quad \Leftrightarrow \quad x_i \text{ not a SV}$$

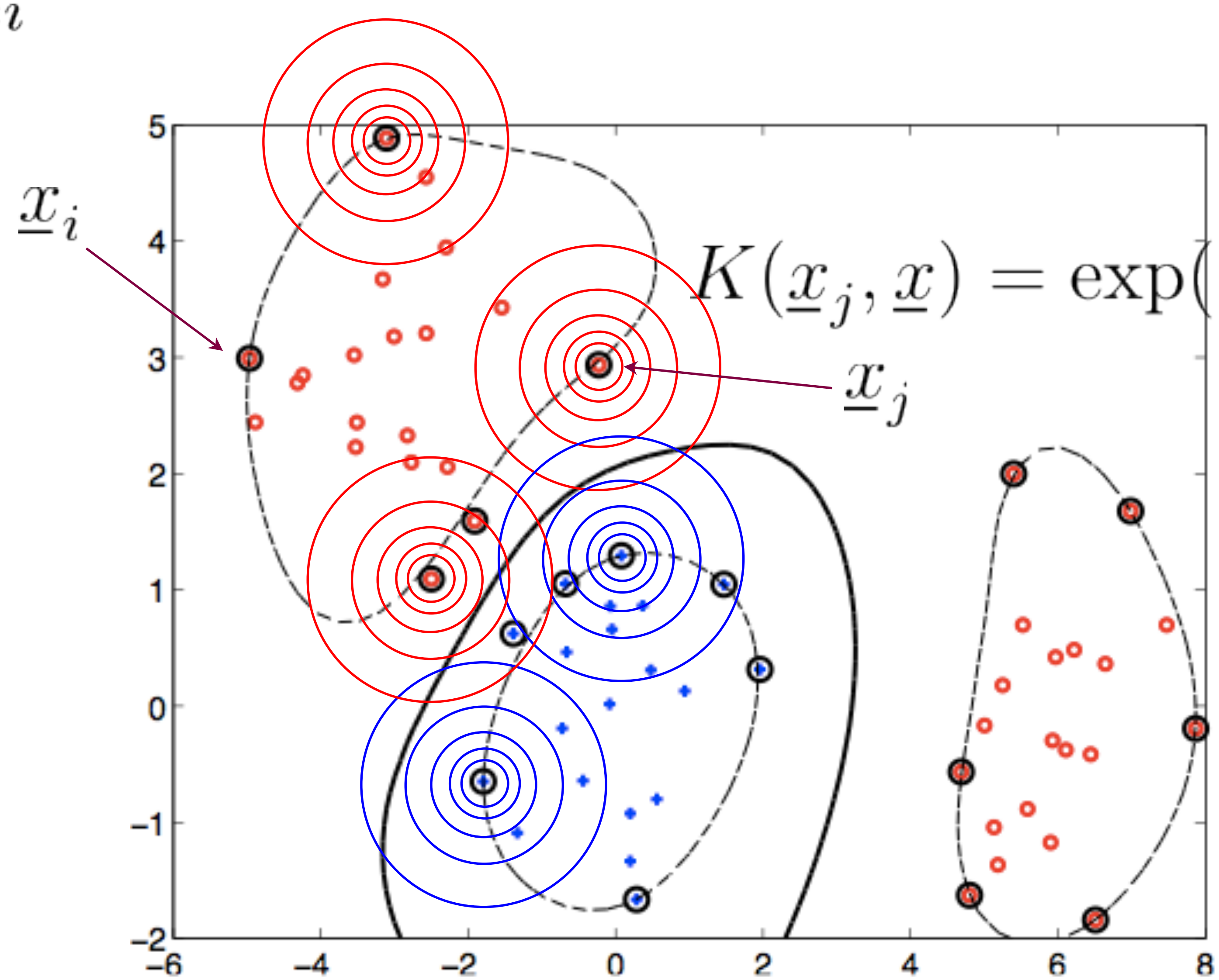


RBF kernel, support vectors

- Assume no offset, no slack. A point is not a support vector if its margin constraint is satisfied (otherwise it has to be a SV)

$$y_i \sum_{j \neq i} \alpha_j y_j K(\underline{x}_j, \underline{x}_i) \geq 1 \quad \Leftrightarrow \quad x_i \text{ not a SV}$$

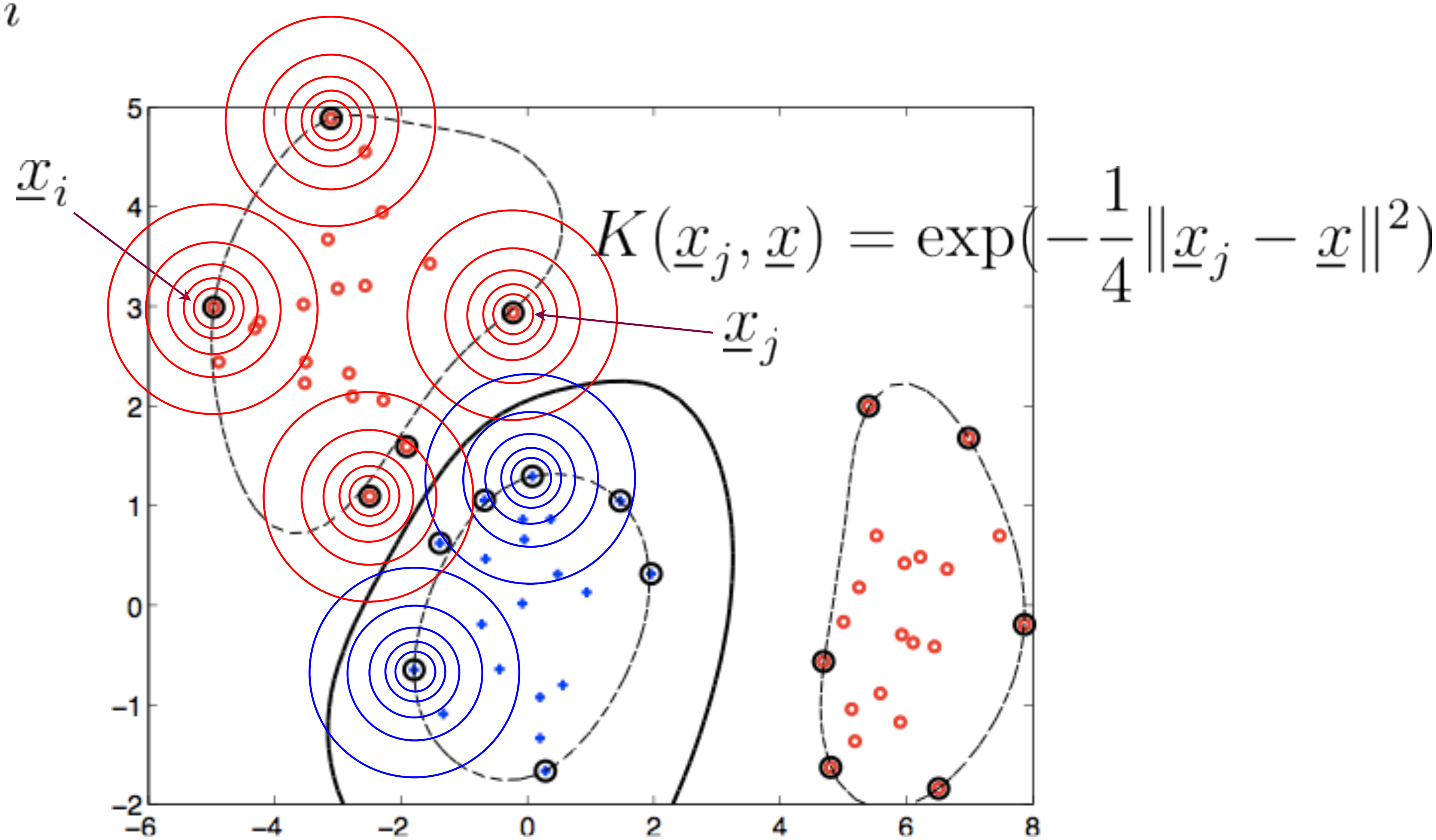
$$K(\underline{x}_j, \underline{x}) = \exp(-\frac{1}{4}\|\underline{x}_j - \underline{x}\|^2)$$



RBF kernel, support vectors

- Assume no offset, no slack. A point is not a support vector if its margin constraint is satisfied (otherwise it has to be a SV)

$$y_i \sum_{j \neq i} \alpha_j y_j K(\underline{x}_j, \underline{x}_i) \geq 1 \quad \Leftrightarrow \quad x_i \text{ not a SV}$$



Dual SVM with offset

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j [\underbrace{\phi(\underline{x}_i) \cdot \phi(\underline{x}_j)}_{\text{kernel}}]$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n, \; \sum_{i=1}^{n} \alpha_i y_i = 0$$

- Where's the offset parameter? How do we solve for it?

Dual SVM with offset

$$\underline{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{[\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x}_j)]}_{\text{kernel}}$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n, \; \sum_{i=1}^{n} \alpha_i y_i = 0$$

- Where's the offset parameter? How do we solve for it?
- We know that the classification constraints are tight for support vectors. If the ith point is a support vector, then

$$y_i(\underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x}_i) + \theta_0^*) = 1$$

Dual SVM with offset

$$\underline{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{[\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x}_j)]}_{\text{kernel}}$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- Where's the offset parameter? How do we solve for it?
- We know that the classification constraints are tight for support vectors. If the ith point is a support vector, then

$$y_i(\underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x}_i) + \theta_0^*) = 1$$

$$\Rightarrow \theta_0^* = y_i - \underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x}_i) = y_i - \sum_{j=1}^{n} \alpha_j^* y_j \underbrace{[\underline{\phi}(\underline{x}_j) \cdot \underline{\phi}(\underline{x}_i)]}_{\text{kernel}}$$

Dual SVM with offset and slack

$$\underline{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j [\underbrace{\phi(\underline{x}_i) \cdot \phi(\underline{x}_j)}_{\text{kernel}}]$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, \; i = 1, \ldots, n, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- C is the same slack penalty as in the primal formulation with slack variables (this dual was derived from that primal).
- To find the SVs, and solve for the offset parameter, we find the points where $0 < \alpha_i < C$
- Points where $\alpha_i = C$ violate the margin constraints.

Outline

- Multi-class classification
  - Approach 1: Multi-class SVM
  - Approach 2: combine the predictions of a set of binary classifiers, via output codes
- Ranking
  - SVM-Rank

Direct multi-class SVM

- We can also try to directly solve the multi-class problem, analogously to binary SVMs
- If there are k classes, we introduce k parameter vectors, one for each class
- We learn the parameters jointly by ensuring that the discriminant function associated with the correct class has the highest value

$$\text{minimize} \quad \frac{1}{2}\sum_{y=1}^{k} \|\underline{\theta}_y\|^2 \quad \text{subject to}$$

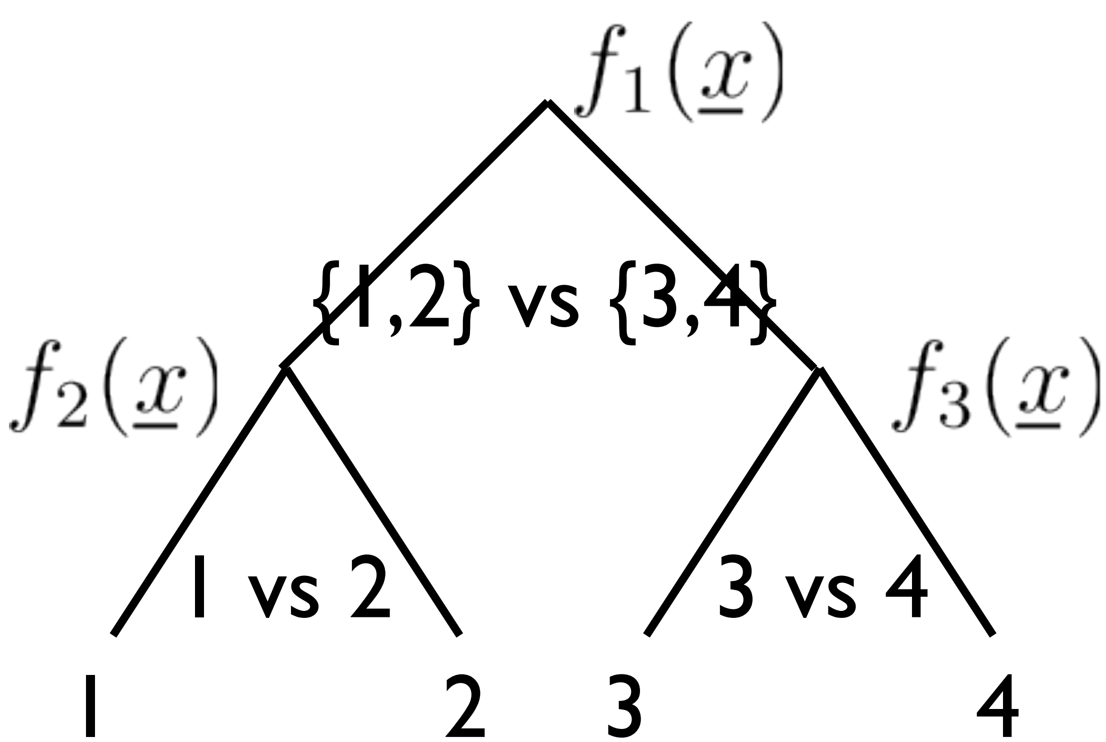
$$(\underline{\theta}_{y_i} \cdot \underline{\phi}(\underline{x}_i)) \geq (\underline{\theta}_{y'} \cdot \underline{\phi}(\underline{x}_i)) + 1, \ \forall y' \neq y_i, \ i = 1, \dots, n$$

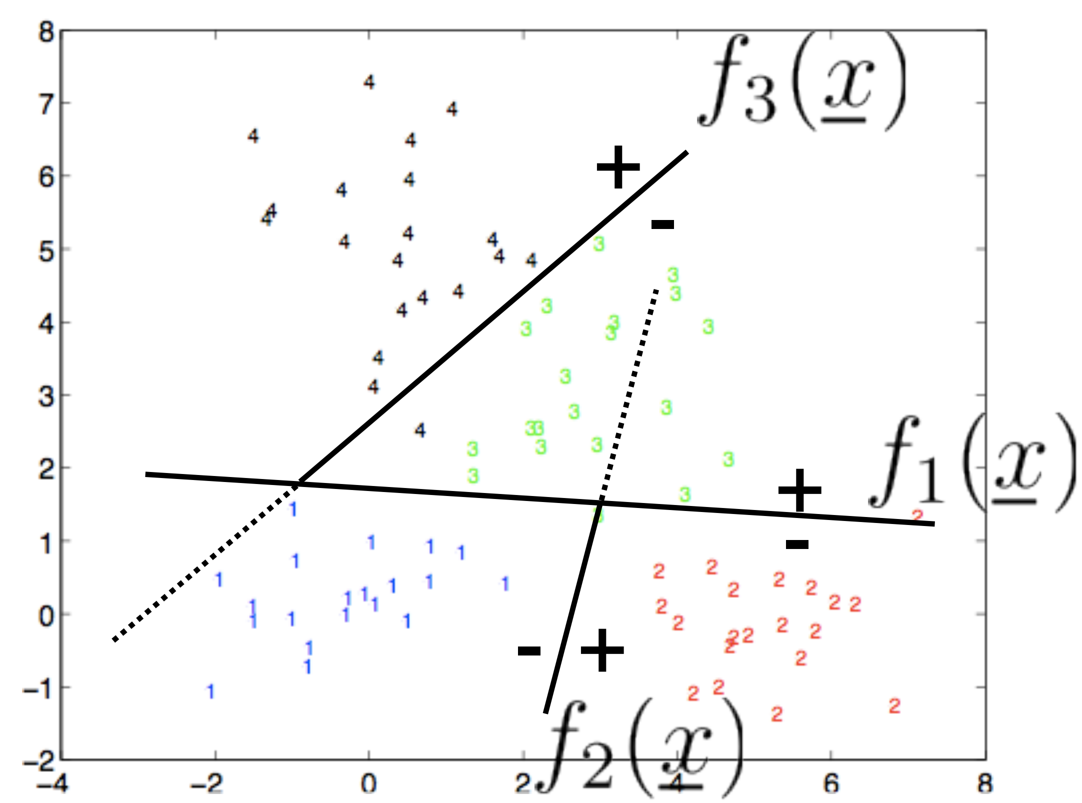
- For new examples, we predict labels according to

$$\hat{y} = \arg\max_{y=1,\dots,k} (\underline{\theta}_y^* \cdot \underline{\phi}(\underline{x}))$$

Multi-way classification

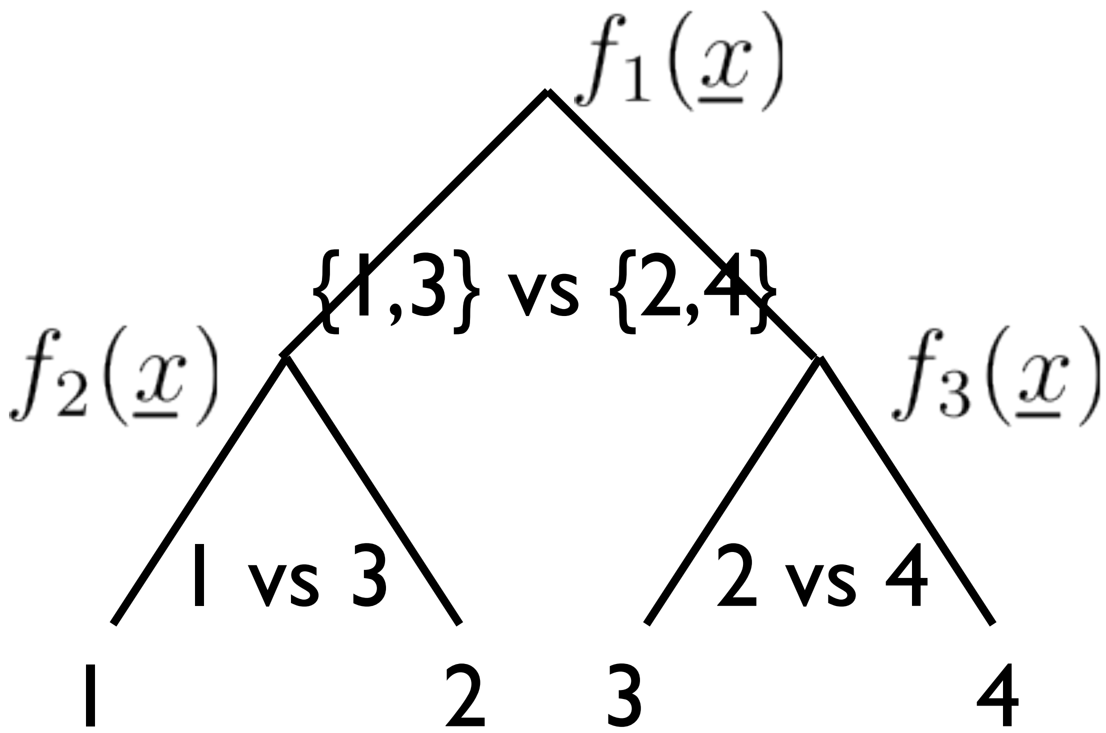
- Character recognition, face recognition, tumor identification, etc., are not binary classification problems
- We can, however, reduce multi-way classification problems to sets of binary classification problems

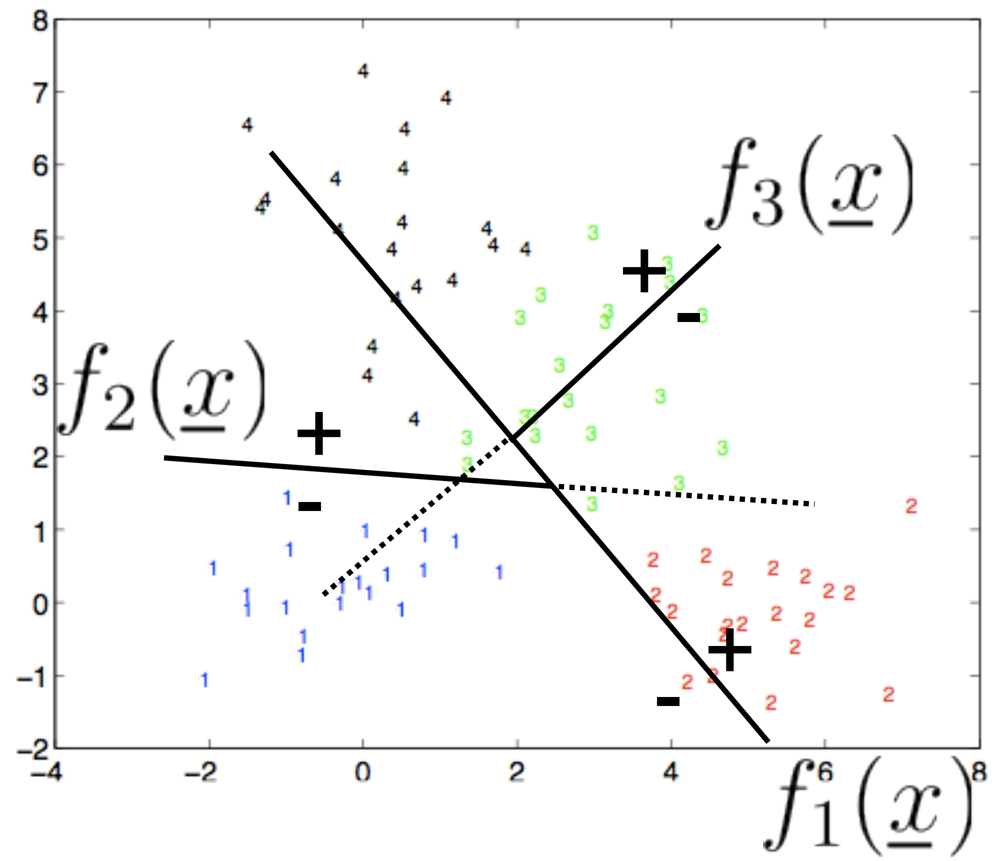




Reducing multi-class to binary

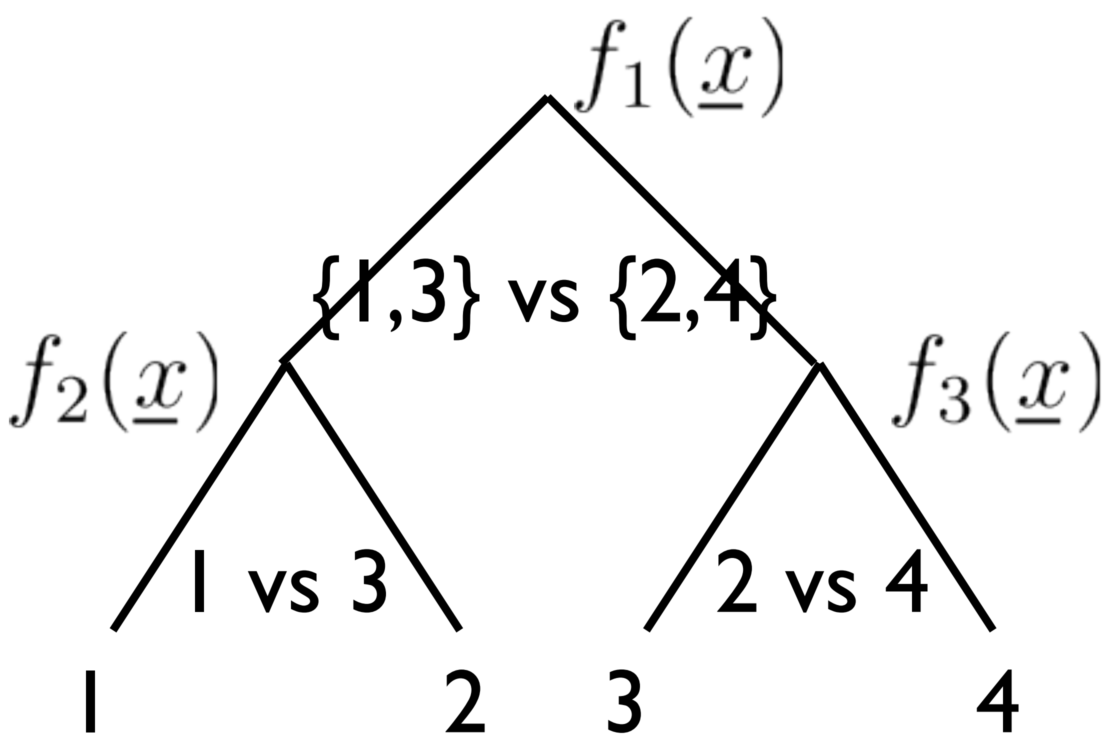
- How we partition the classes into binary problems matters a great deal

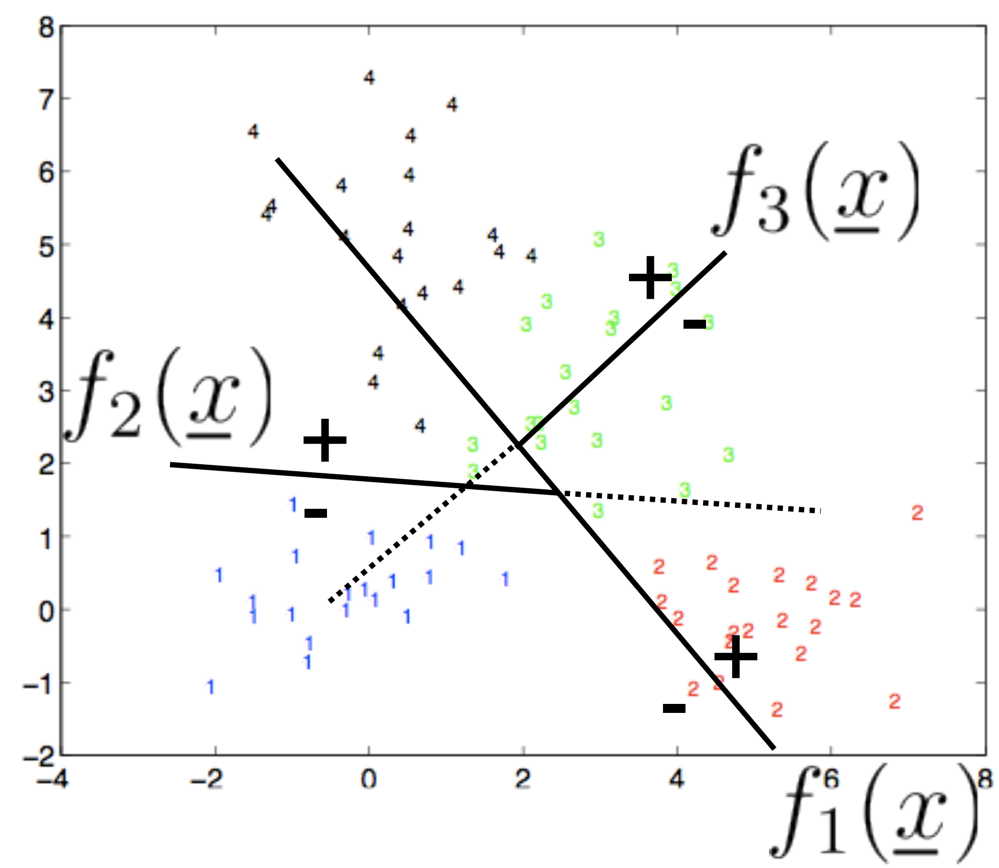




Reducing multi-class to binary

- How we partition the classes into binary problems matters a great deal





- Things to consider
  - accuracy we can achieve for each binary task
  - redundancy of the binary tasks
  - cost of using many binary classifiers

Reducing multi-class to binary

- We can think of each partitioning scheme as defining an "output code" matrix where rows correspond to multi-way labels and columns specify binary classification tasks

binary tasks

$$
\text{classes} \quad
\begin{array}{c c}
 & \begin{array}{ccc} f_1(\underline{x}) & f_2(\underline{x}) & f_3(\underline{x}) \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}
\end{array}
$$

{1,3} vs {2,4}   1 vs 3   2 vs 4

- The binary classifiers are trained independently of each other

Reducing multi-class to binary

- We can think of each partitioning scheme as defining an “output code” matrix where rows correspond to multi-way labels and columns specify binary classification tasks

binary tasks

$$
\text{classes}\quad
\begin{array}{c c}
 & \begin{array}{ccc} f_1(\underline{x}) & f_2(\underline{x}) & f_3(\underline{x}) \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} &
\begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}
\end{array}
$$

the 3rd classifier sees any training example from class 4 as being labeled +1

{1,3} vs {2,4}   1 vs 3   2 vs 4

- The binary classifiers are trained independently of each other

Reducing multi-class to binary

- We can think of each partitioning scheme as defining an "output code" matrix where rows correspond to multi-way labels and columns specify binary classification tasks

binary tasks

$$
\text{classes} \quad
\begin{array}{c}
 \\ 1 \\ 2 \\ 3 \\ 4
\end{array}
\begin{array}{c}
f_1(\underline{x}) \; f_2(\underline{x}) \; f_3(\underline{x}) \\
\left[\begin{array}{ccc}
-1 & -1 & 0 \\
1 & 0 & -1 \\
-1 & 1 & 0 \\
1 & 0 & 1
\end{array}\right]
\end{array}
$$

the 3rd classifier is not trained with examples from class 1

the 3rd classifier sees any training example from class 4 as being labeled +1

{1,3} vs {2,4}   1 vs 3   2 vs 4

- The binary classifiers are trained independently of each other

Reducing multi-class to binary

- We can think of each partitioning scheme as defining an "output code" matrix where rows correspond to multi-way labels and columns specify binary classification tasks

binary tasks

$$\begin{array}{cc} & \begin{array}{cccc} f_1(\underline{x}) & f_2(\underline{x}) & f_3(\underline{x}) & f_4(\underline{x}) \end{array} \\ \text{classes} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix} \end{array}$$

4th classifier sees any training example from class 3 as being labeled -1

One-versus-all output code matrix R

Reducing multi-class to binary

- We can think of each partitioning scheme as defining an "output code" matrix where rows correspond to multi-way labels and columns specify binary classification tasks

$$
\text{classes}\quad
\begin{array}{c}
1\\ 2\\ 3\\ 4
\end{array}
\overset{\text{binary tasks}}{\begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 1 & 1 & 0 \\
0 & -1 & 0 & -1 & 0 & 1 \\
0 & 0 & -1 & 0 & -1 & -1
\end{bmatrix}}
$$

All-pairs output code matrix R

- Properties of good code matrices
  - "binary codes" (rows) should be well-separated (good error correction)
  - binary tasks (columns) should be easy to solve

Output codes, error correction

- A generalized hamming distance between "code words" (rows of the output code matrix)

$$\Delta(y, y') = \sum_{j=1}^{m} \frac{1 - R(y,j)R(y',j)}{2}$$

- Row separation $\rho = \min_{y \neq y'} \Delta(y, y')$

$$\begin{array}{cc} & \text{m binary tasks} \\ \begin{array}{cc} & 1 \\ & 2 \\ \text{classes y} & 3 \\ & 4 \end{array} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix} \end{array}$$

Reducing multi-class to binary

- Predicting the label for a new example consists of finding the row of the code matrix most consistent with the binary predictions (of the discriminant functions)

$$
\begin{array}{cc}
 & \text{binary tasks } j \\
\begin{array}{c} \\ \\ \text{classes } y \\ \\ \end{array}
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} &
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 1 & 1 & 0 \\
0 & -1 & 0 & -1 & 0 & 1 \\
0 & 0 & -1 & 0 & -1 & -1
\end{bmatrix}
\end{array}
$$

$$
\hat{y} = \operatorname*{argmin}_{y} \sum_{j=1}^{m} \text{Loss}\Big(\overbrace{\underbrace{R(y,j)}_{\substack{\text{target binary label for} \\ \text{the jth classifier if} \\ \text{the multi-class label is y}}} \, \underbrace{\hat{\underline{\theta}}_j \cdot \phi(\underline{x})}_{\substack{\text{discriminant function value} \\ \text{of the jth classifier in response to} \\ \text{the new example}}}}^{\text{agreement}} \Big)
$$

Output codes, error correction

- If the loss is the hinge loss, loss(z) = max(0,1-z), then on n training examples,

$$\#\ \text{multi-class errors on the training set}$$

$$\leq \frac{1}{\rho} \sum_{t=1}^{n} \left[\sum_{j=1}^{m} \text{Loss}\big(\, R(y_t, j)\, \underline{\hat{\theta}}_j \cdot \underline{\phi}(\underline{x}_t) \,\big) \right]$$

small if code words are well-separated

small if each binary task can be solved well