CS578 Statistical Machine Learning Lecture 5

Jean Honorio
Purdue University

(based on slides by Tommi Jaakkola, MIT CSAIL)

Today's topics

- Brief review
  - support vector machine with kernels
- One-class problems, anomaly detection
  - simple formulation, dual
  - removing outliers
- Multi-way classification
  - reducing multi-class to binary
  - margin based solution

- **Homework 1**: due Jan 30, 11.59pm EST. ***MATLAB only***
- **Office hours:** see webpage

Dual SVM

- Select the kernel and penalty C, then solve

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

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

- Support vectors (SV) are identified by $\alpha_i^* > 0$
- Solve θ_0^* based on tight constraints $\alpha_i^* > 0$
- Predict labels for new points according to

$$f(\underline{x}; \alpha^*) \;\; = \;\; \text{sign}\Big(\sum_{i \in SV} \alpha_i^* y_i K(\underline{x}_i, \underline{x}_j) + \theta_0^* \Big)$$

Today's topics

- Brief review
  - support vector machine with kernels
- One-class problems, anomaly detection
  - simple formulation, dual
  - removing outliers
- Multi-way classification
  - reducing multi-class to binary
  - margin based solution

Anomaly detection

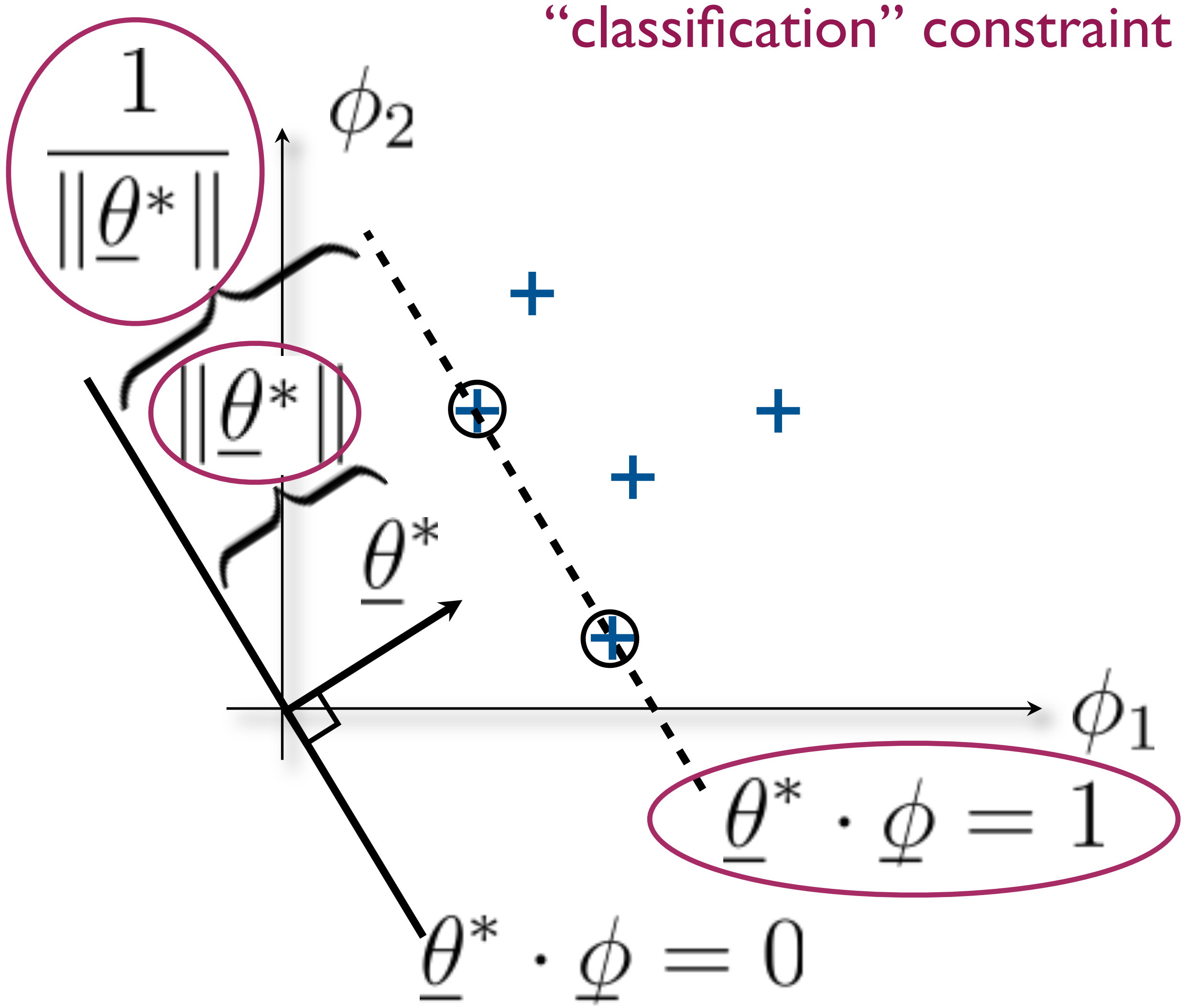
- In anomaly detection, we wish to identify examples that are not part of the "positive" class
  - monitoring, intrusion detection, fault detection, retrieval applications, etc.
- The goal is to learn a separator that "envelopes" the typical (positive) examples, enabling us to rank how "positive" each example is
- We can formulate the estimation problem without access to any negative examples (that may be hard to come by, or too diverse to model well)

Linear separation from origin

- A simple formulation of separating a set of positive examples from the origin (in the feature space)

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

$$\text{subject to } \underbrace{\underline{\theta} \cdot \underline{\phi}(\underline{x}_i) \geq 1}_{\text{“classification” constraint}}, \; i = 1, \ldots, n$$

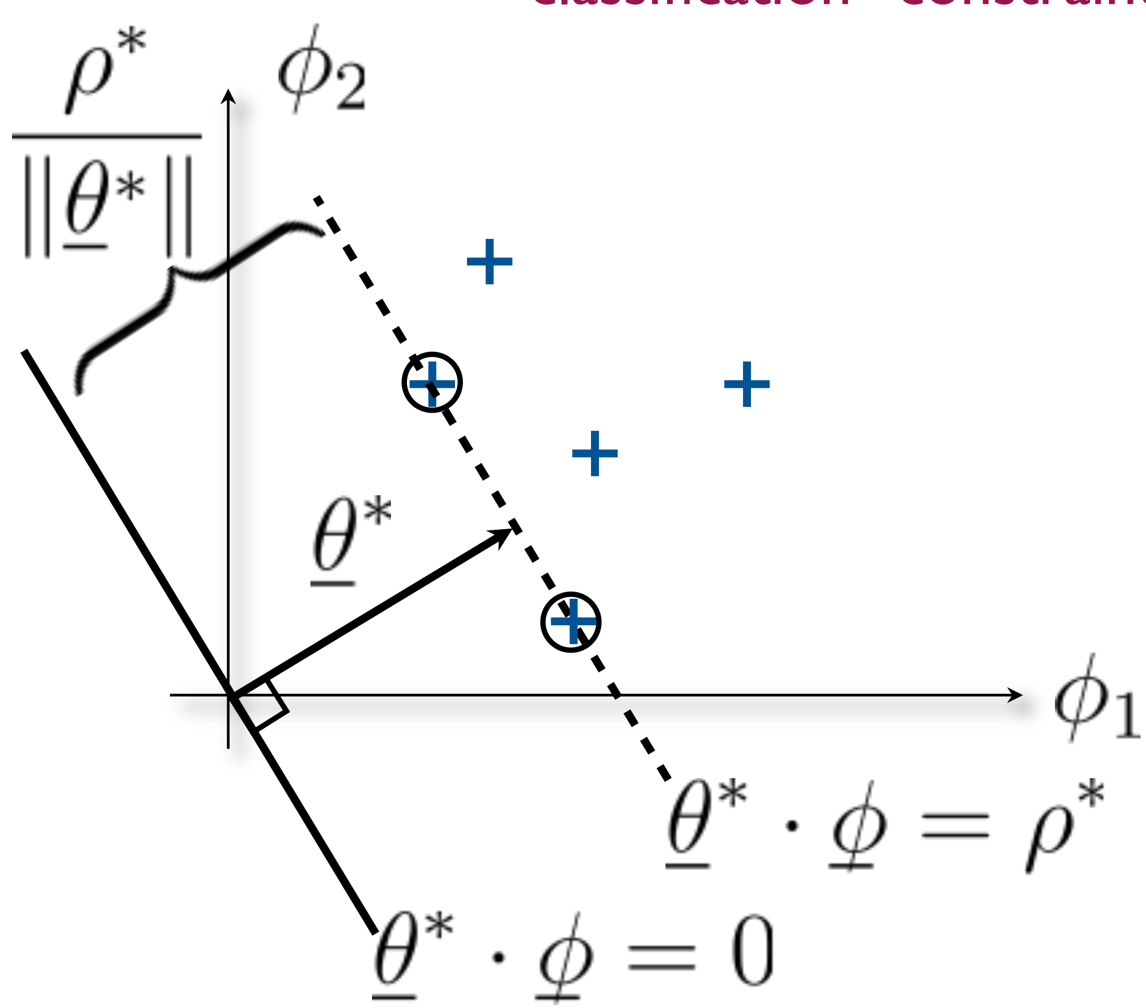


Linear separation from origin

- A simple formulation of separating a set of positive examples from the origin (in the feature space)

$$\text{minimize } \frac{1}{2}\|\underline{\theta}\|^2 - \rho \text{ with respect to } \underline{\theta},\ \rho$$

$$\text{subject to } \underbrace{\underline{\theta}\cdot\underline{\phi}(\underline{x}_i) \geq \rho}_{\text{“classification” constraint}},\ i = 1,\ldots,n$$

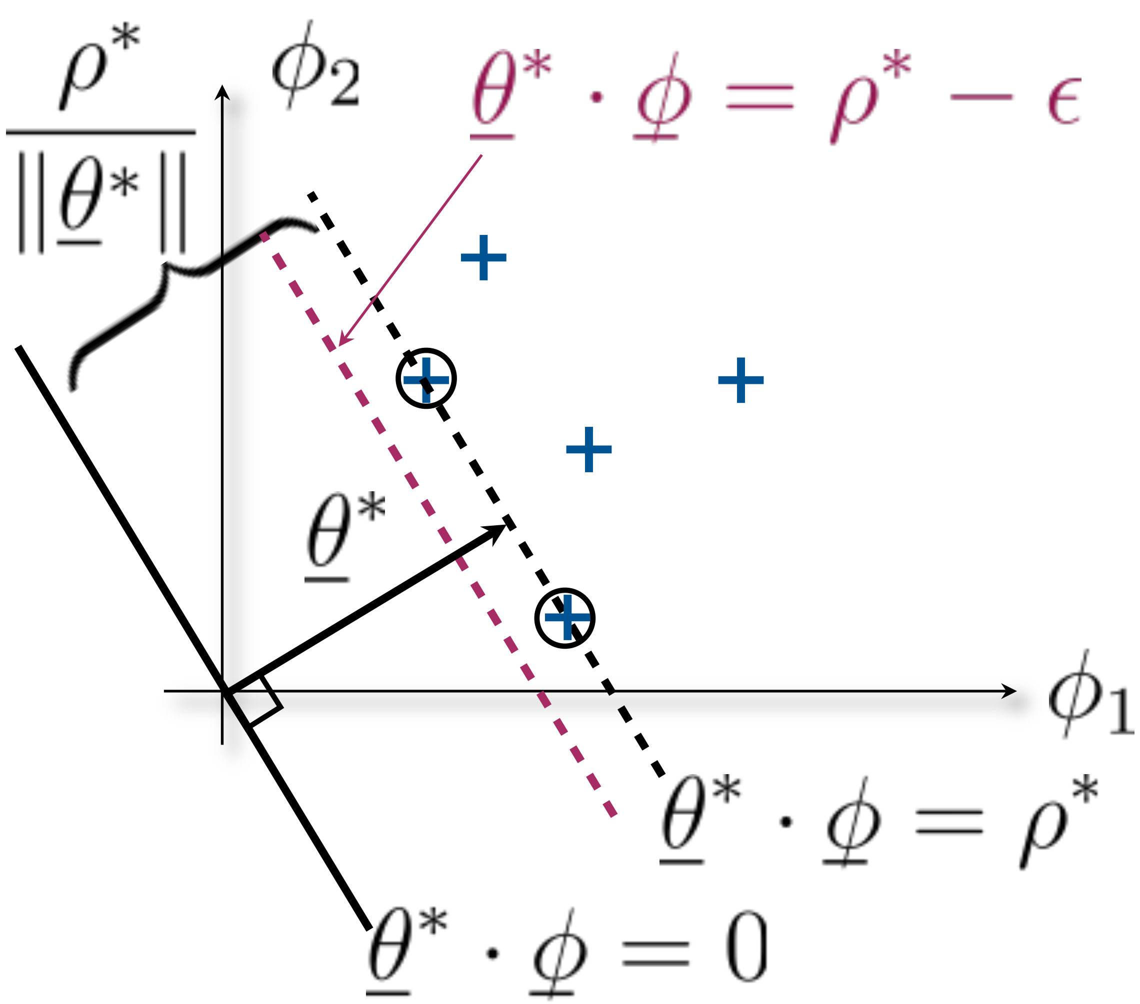


Linear separation from origin

- A simple formulation of separating a set of positive examples from the origin (in the feature space)

$$\text{minimize } \frac{1}{2}\|\underline{\theta}\|^2 - \rho \text{ with respect to } \underline{\theta}, \rho$$

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

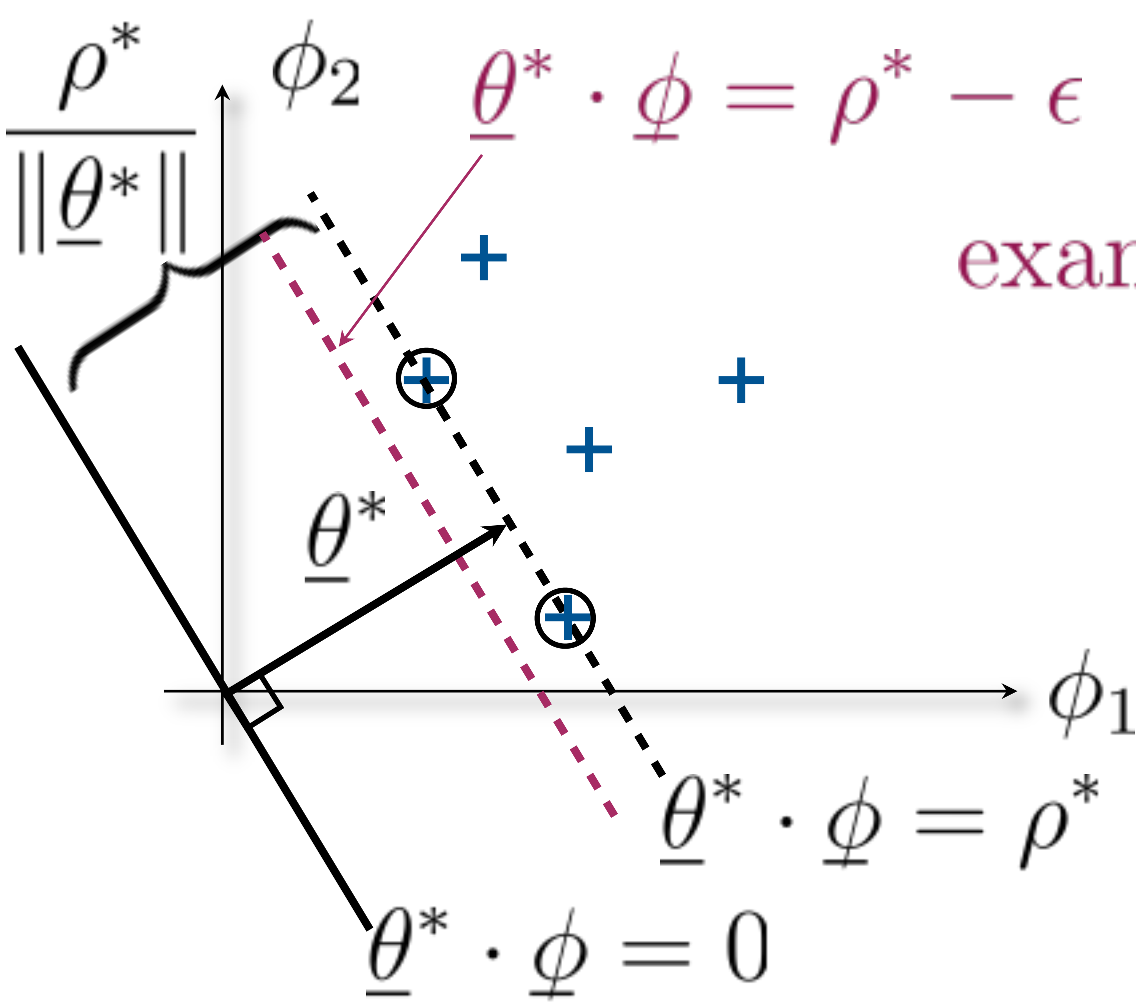


Linear separation from origin

- A simple formulation of separating a set of positive examples from the origin (in the feature space)

$$\text{minimize} \quad \frac{1}{2}\|\underline{\theta}\|^2 - \rho \quad \text{with respect to } \underline{\theta},\ \rho$$

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



example is "typical" if

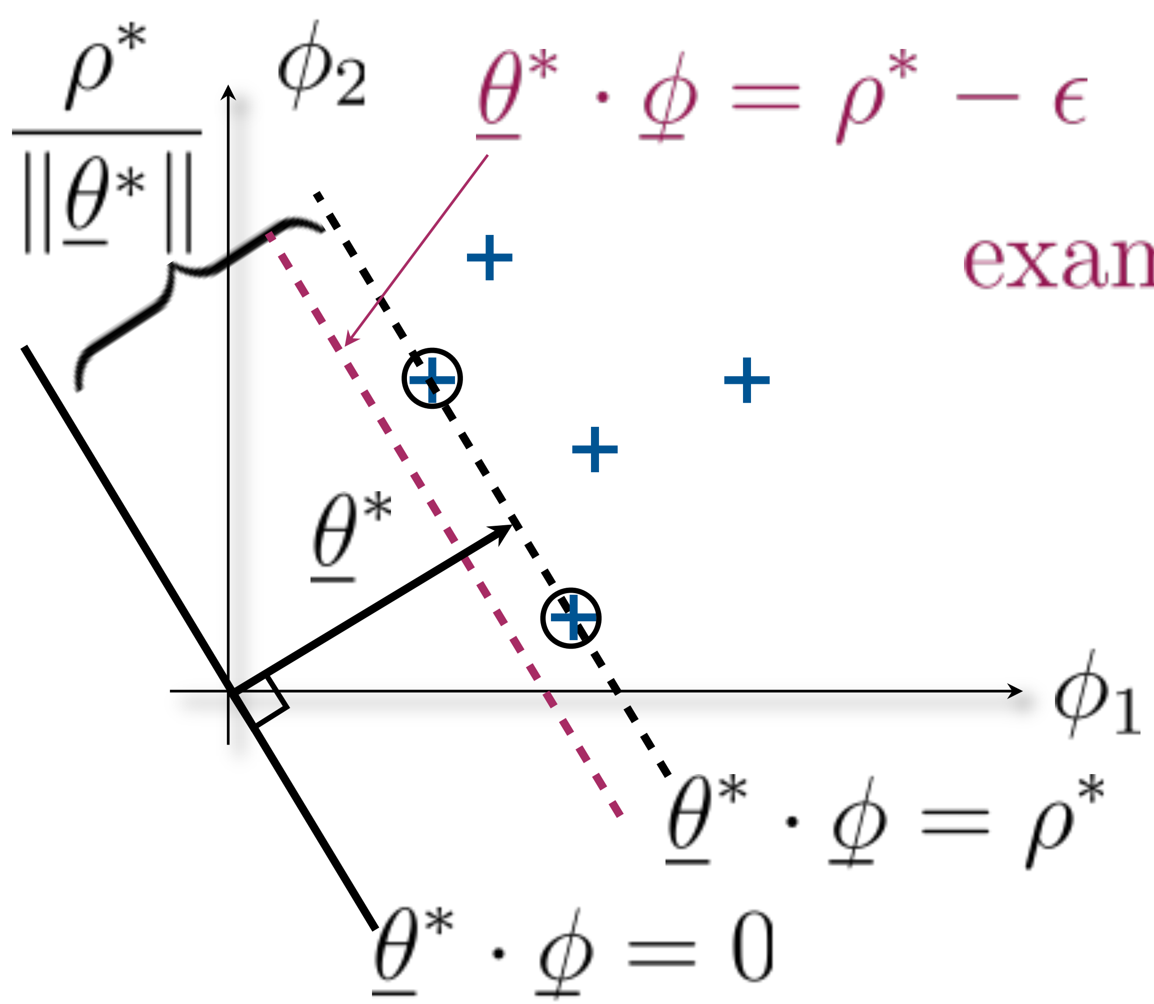
$$\underline{\theta}^* \cdot \underline{\phi}(\underline{x}) \geq \rho^* - \epsilon$$

Linear separation from origin

- A simple formulation of separating a set of positive examples from the origin (in the feature space)

$$\text{minimize} \quad \frac{1}{2}\|\underline{\theta}\|^2 - \rho \quad \text{with respect to } \underline{\theta},\ \rho$$

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



example is "typical" if

$$\underline{\theta}^* \cdot \underline{\phi}(\underline{x}) \geq \rho^* - \epsilon$$

the relaxed threshold
could be set via cross-validation

Dual problem

- The dual problem can be obtained analogously to support vector machines

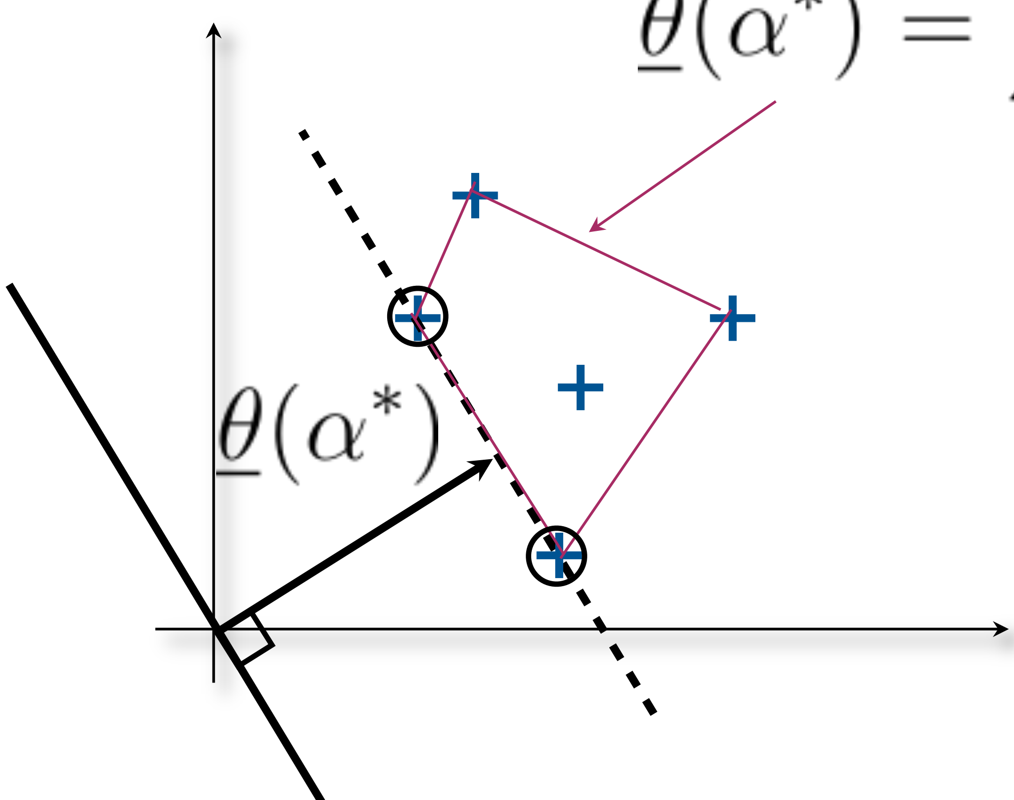
$$\text{maximize} \quad -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j[\underline{\phi}(\underline{x}_i)\cdot\underline{\phi}(\underline{x}_j)]$$

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

where

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

convex hull of the feature vectors



Dual problem

- The dual problem can be obtained analogously to support vector machines

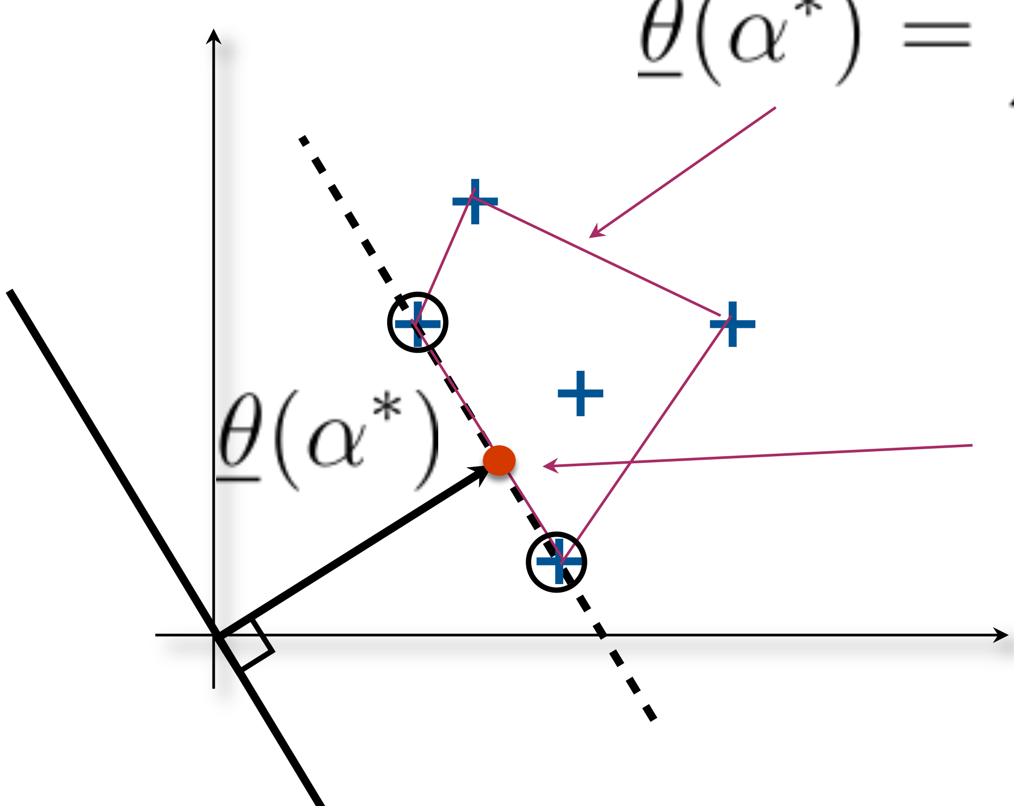
$$\text{maximize} \quad -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j[\underline{\phi}(\underline{x}_i)\cdot\underline{\phi}(\underline{x}_j)]$$

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

where

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

convex hull of the feature vectors



the point within the convex hull of the feature vectors that is closest to the origin

Dual problem

- The dual problem can be obtained analogously to support vector machines

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

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

where

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

- At least one constraint $\underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x}_i) \geq \rho^*$ is tight, so

$$\rho^* = \min_j \underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x}_j) = \min_j \sum_{i=1}^{n} \alpha_i^* [\underline{\phi}(\underline{x}_i) \cdot \underline{\phi}(\underline{x}_j)]$$

Note: similar to our trick for finding the offset θ_0 in Lecture 4

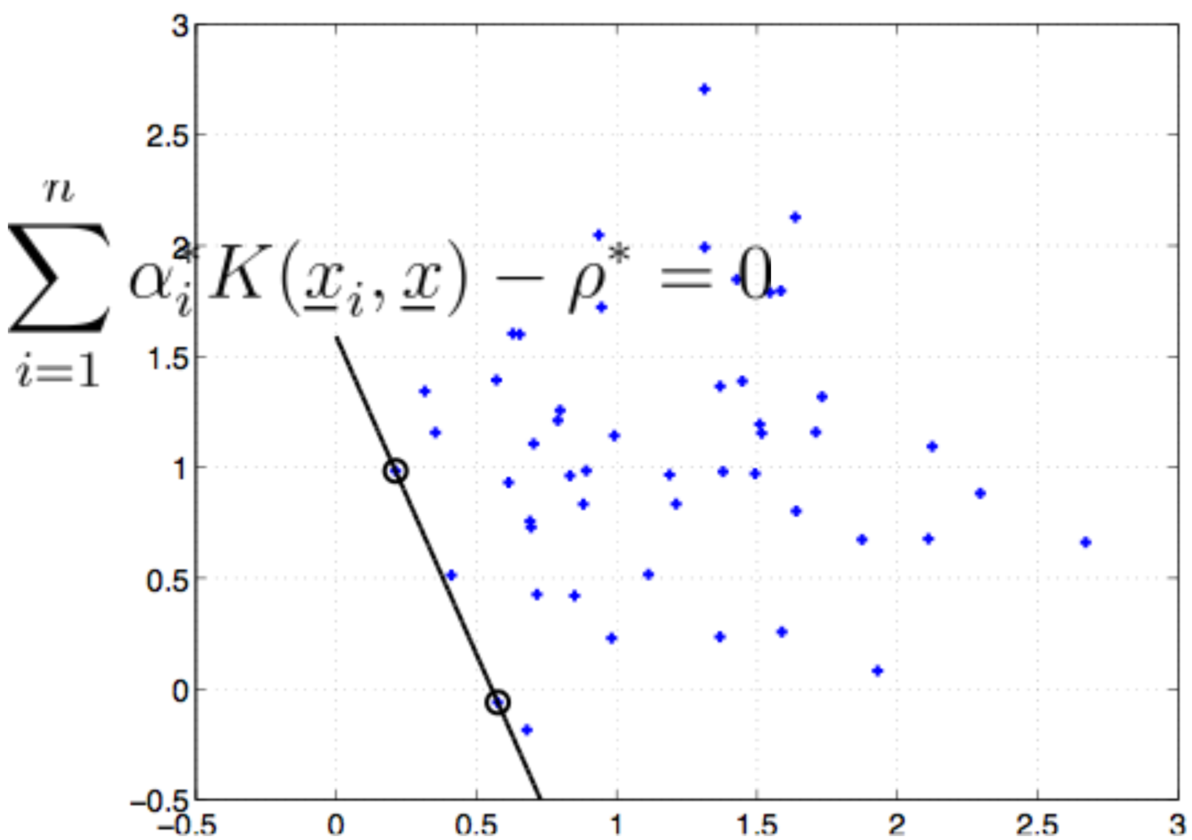
Anomaly detection: examples

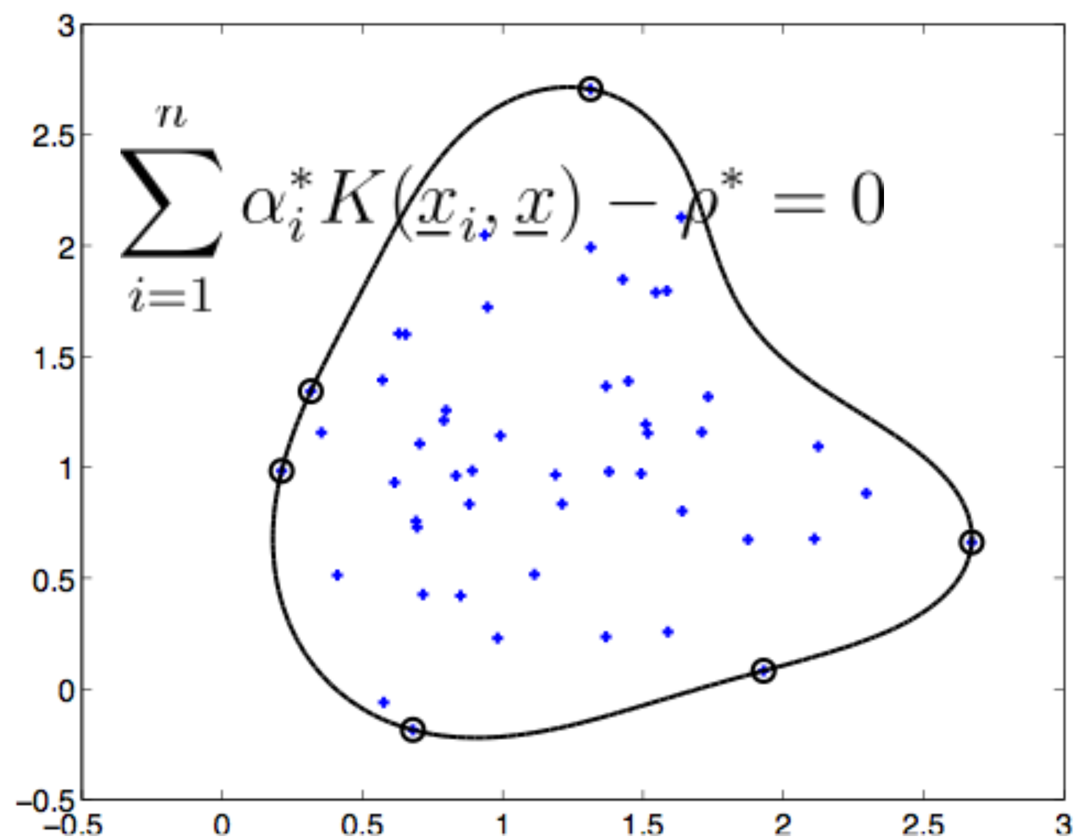
- Examples of margin boundaries

$$\underline{\theta}(\alpha^*) \cdot \underline{\phi}(\underline{x}) - \rho^* = \sum_{i=1}^{n} \alpha_i^* K(\underline{x}_i, \underline{x}) - \rho^* = 0$$

arising from using different kernels (linear, radial basis)

$$K(\underline{x}, \underline{x}') = \underline{x} \cdot \underline{x}' \qquad K(\underline{x}, \underline{x}') = \exp(-1/2\, \|\underline{x} - \underline{x}'\|^2)$$



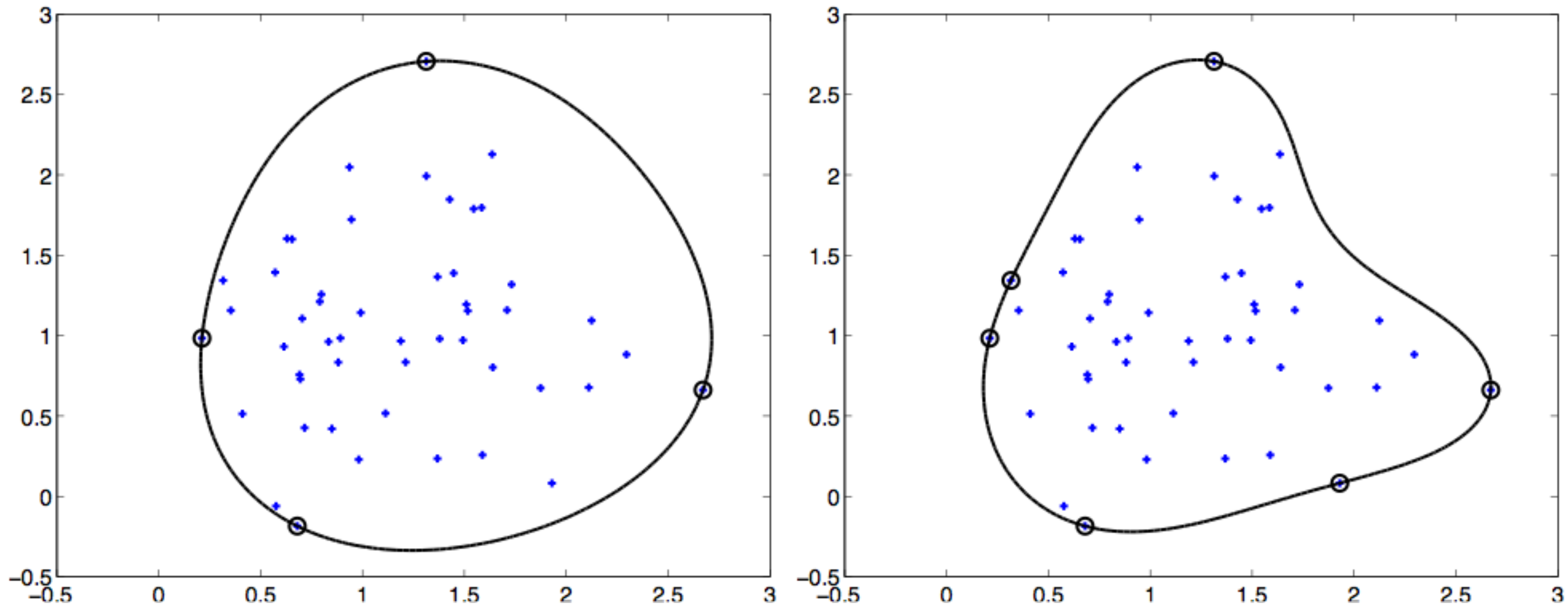


Anomaly detection: examples

- It is important to set the scale parameter correctly in the radial basis kernel

$$K(\underline{x}, \underline{x}') = \exp(-\beta \|\underline{x} - \underline{x}'\|^2),\ \beta > 0$$

$\beta = 1/4$ $\qquad\qquad$ $\beta = 1/2$

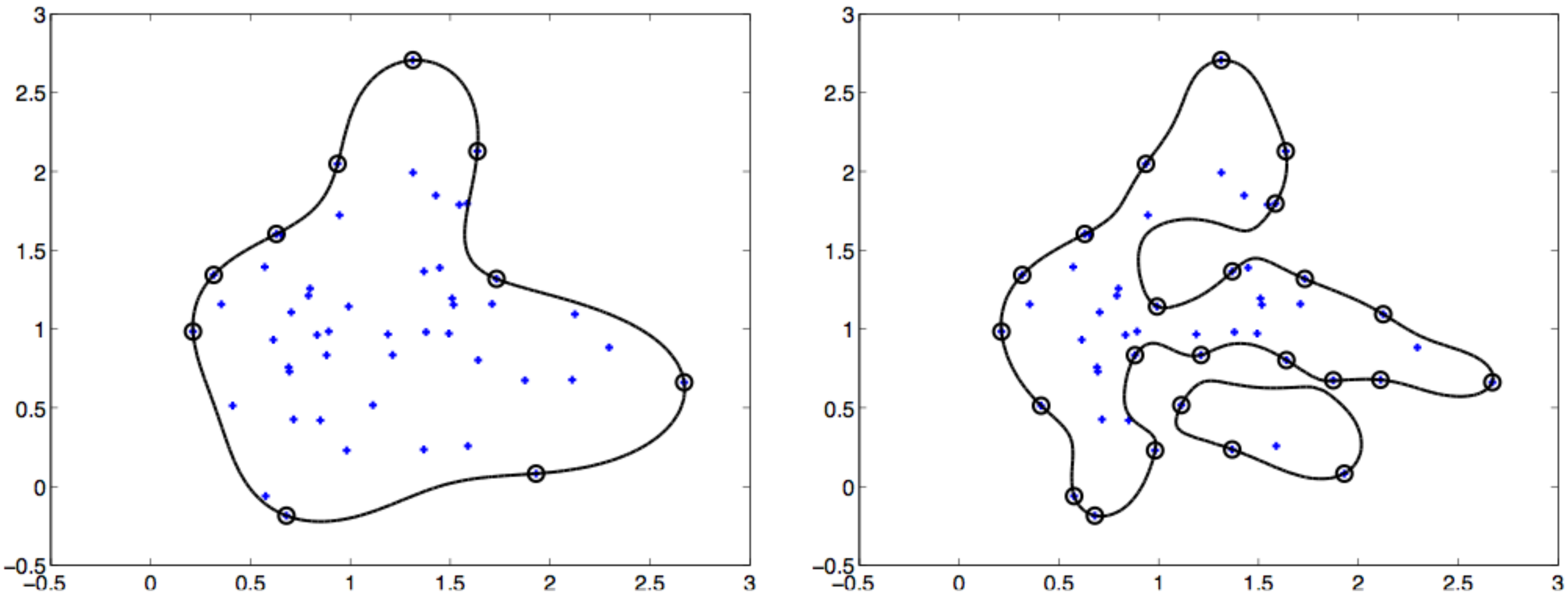


Anomaly detection: examples

- It is important to set the scale parameter correctly in the radial basis kernel

$$K(\underline{x}, \underline{x}') = \exp(-\beta \|\underline{x} - \underline{x}'\|^2), \; \beta > 0$$

$\beta = 1$ $\beta = 4$



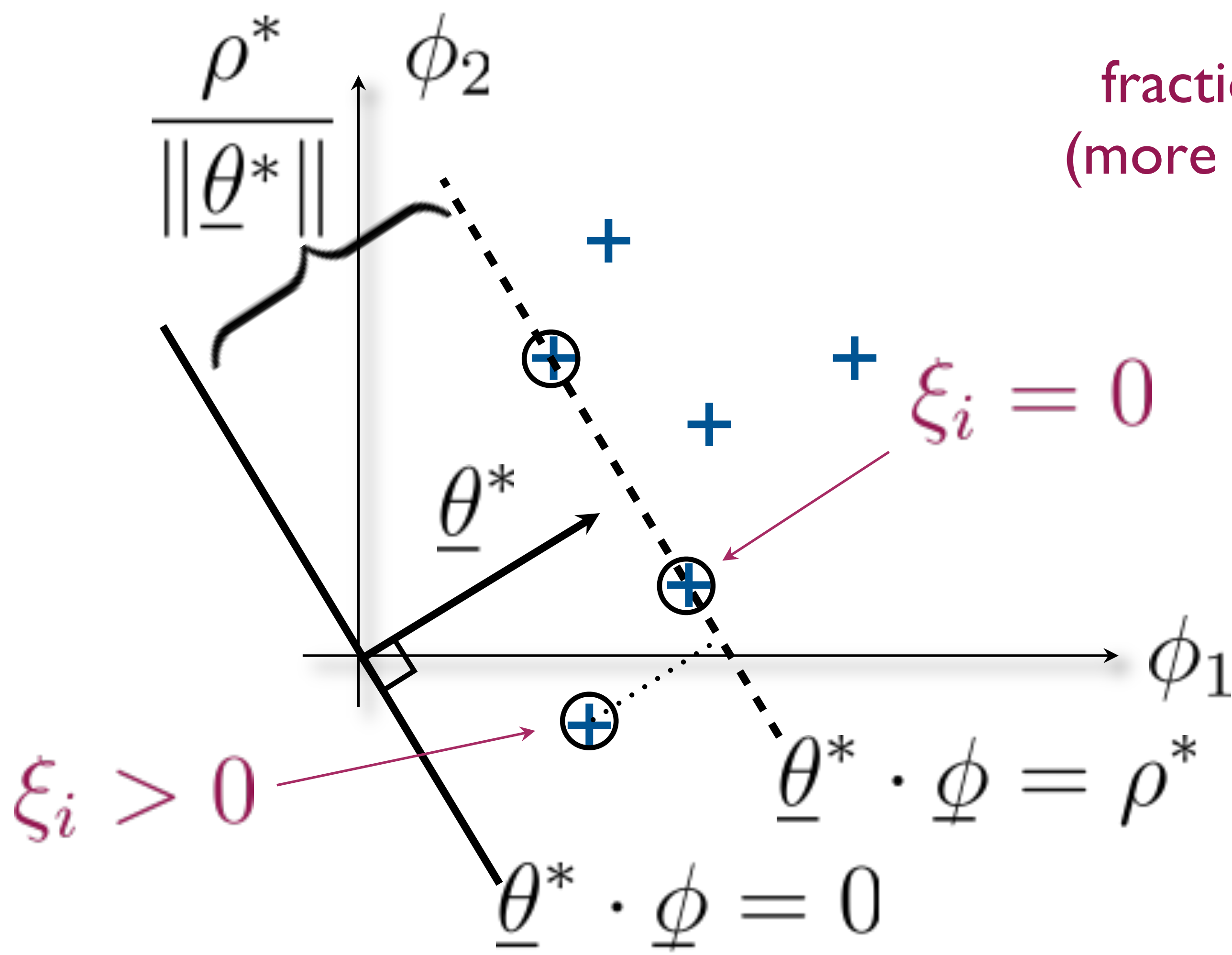
Omitting "outliers"

- We can modify the basic formulation so as to leave a specific fraction of examples at/outside the margin

$$\text{minimize} \;\; \frac{1}{2}\|\underline{\theta}\|^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \;\; \text{subject to}$$

$$\underline{\theta}\cdot\underline{\phi}(\underline{x}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

fraction we can omit
(more clear in the dual)

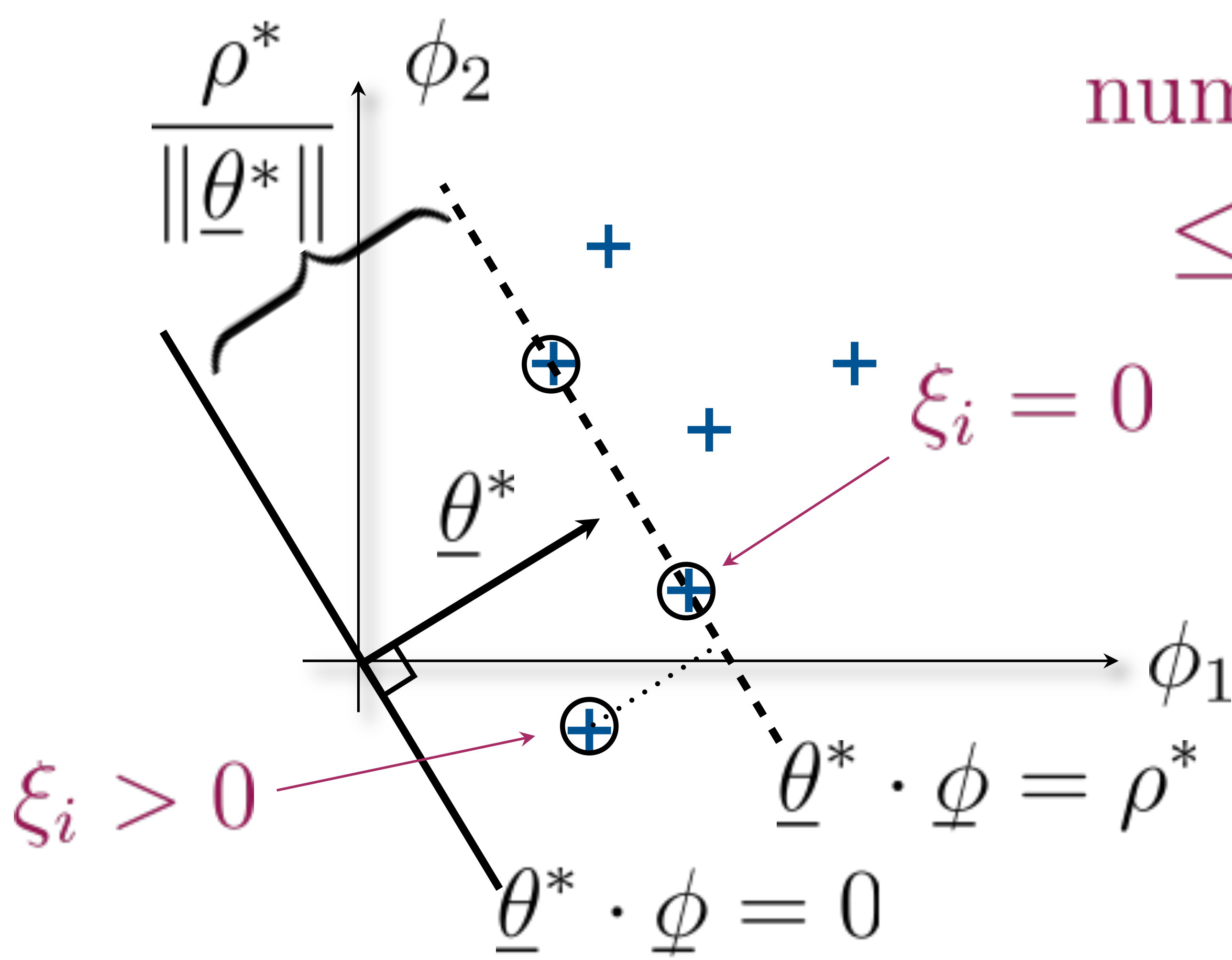


Omitting “outliers”

- We can modify the basic formulation so as to leave a specific fraction of examples at/outside the margin

$$\text{minimize} \quad \frac{1}{2}\|\underline{\theta}\|^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \quad \text{subject to}$$

$$\underline{\theta}\cdot\underline{\phi}(\underline{x}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1,\ldots,n$$



number of non-zero ξ_i’s $\leq \nu\, n \leq$ number of SVs

(See Proposition 1 in [1] if interested.)

Dual problem

- The dual problem can be obtained analogously

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

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

where again

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

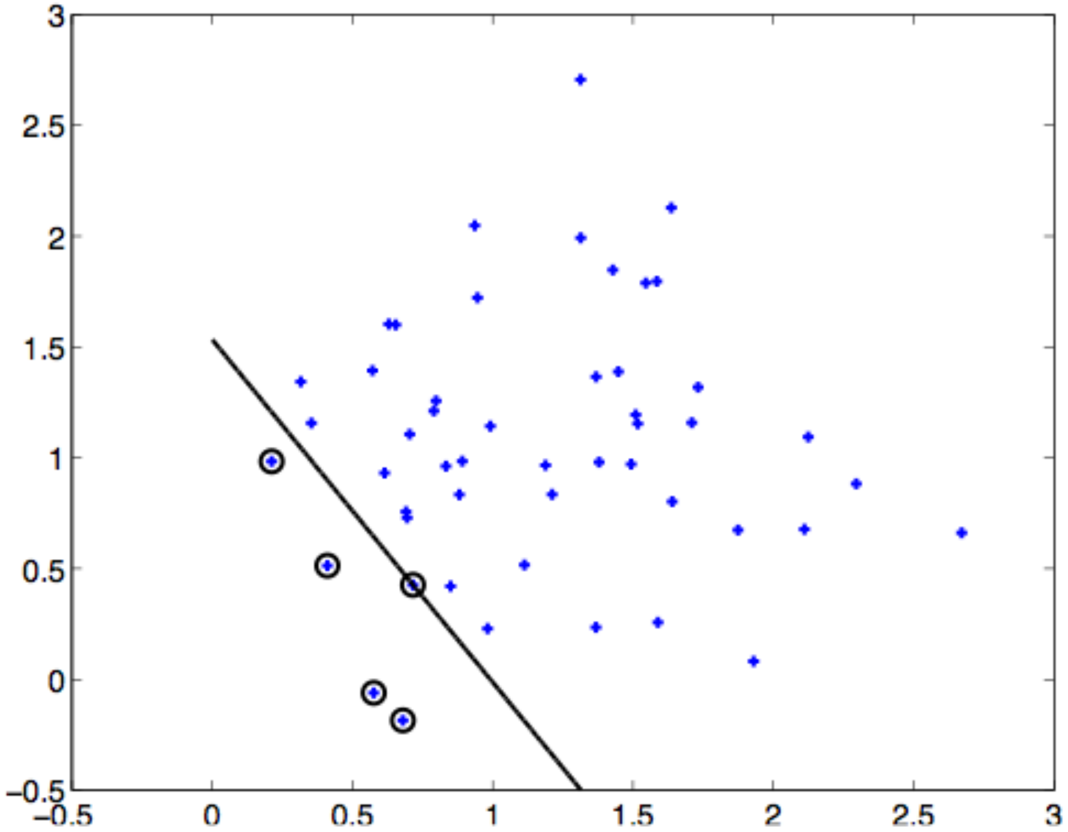
if ν=1, then solution is
α_i^* = 1/*n*, for all *i*

- ρ^* can be estimated from tight constraints with zero slack, i.e., those corresponding to $\alpha_i^* > 0$

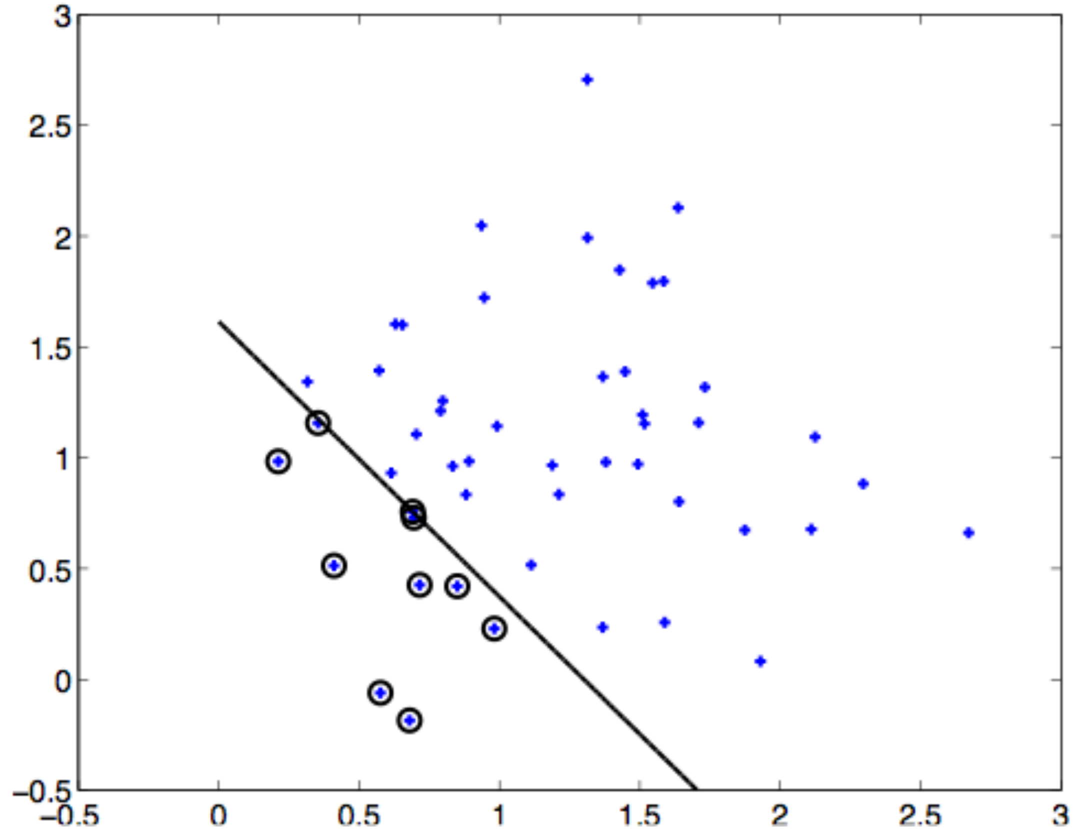
Examples

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

$\nu = 0.1$



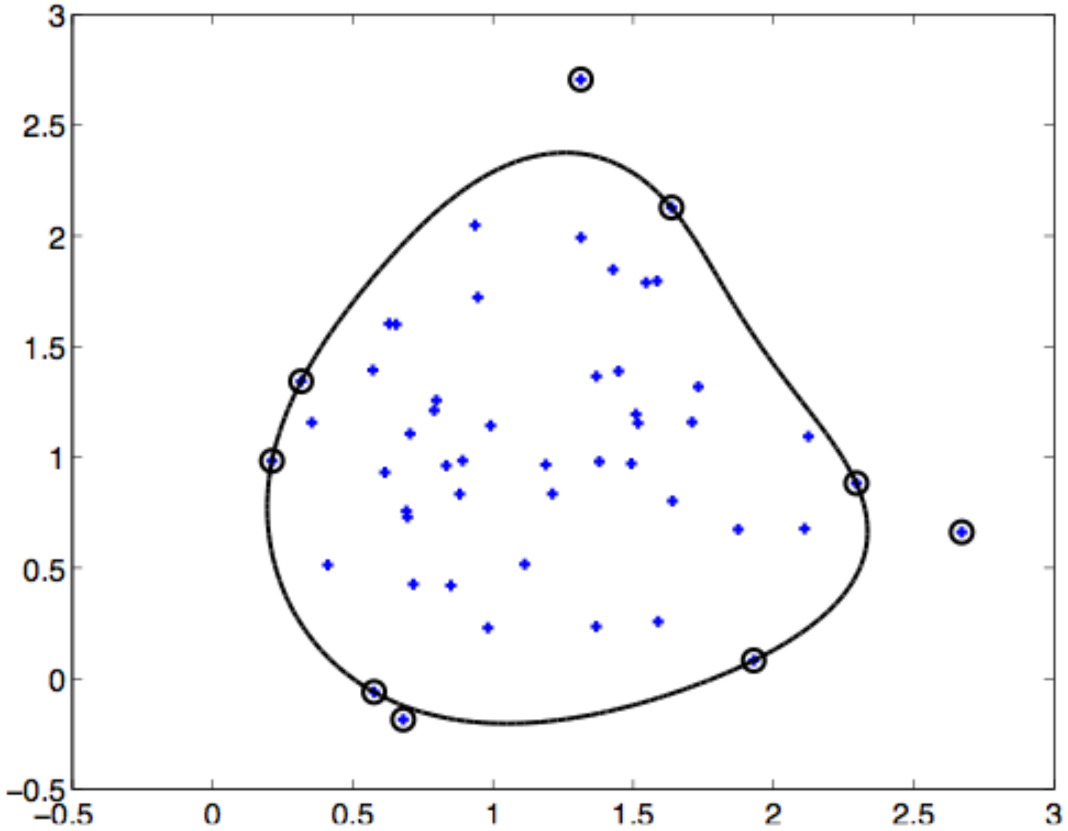
$\nu = 0.2$



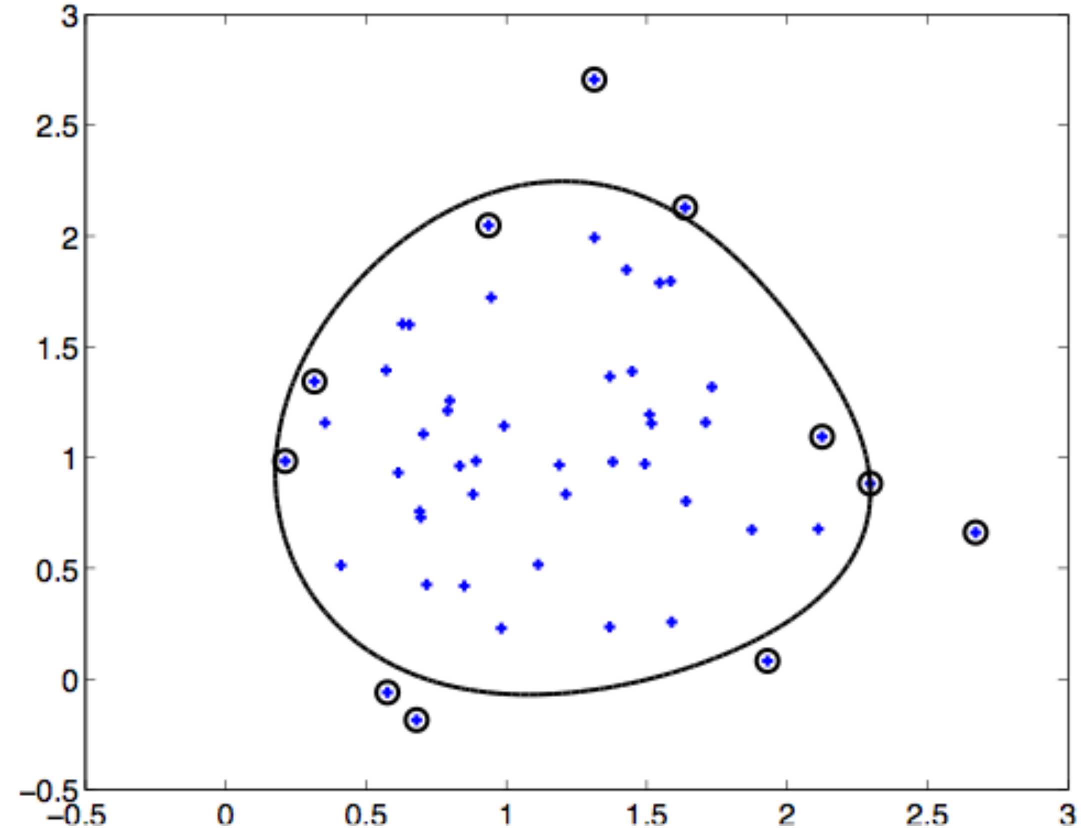
Examples

$$K(\underline{x}, \underline{x}') = \exp(-1/2\|\underline{x} - \underline{x}'\|^2)$$

$\nu = 0.1$



$\nu = 0.2$

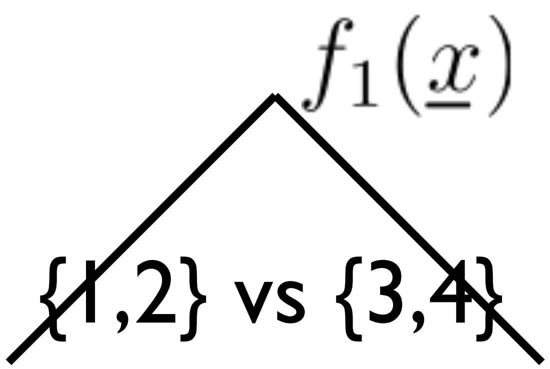


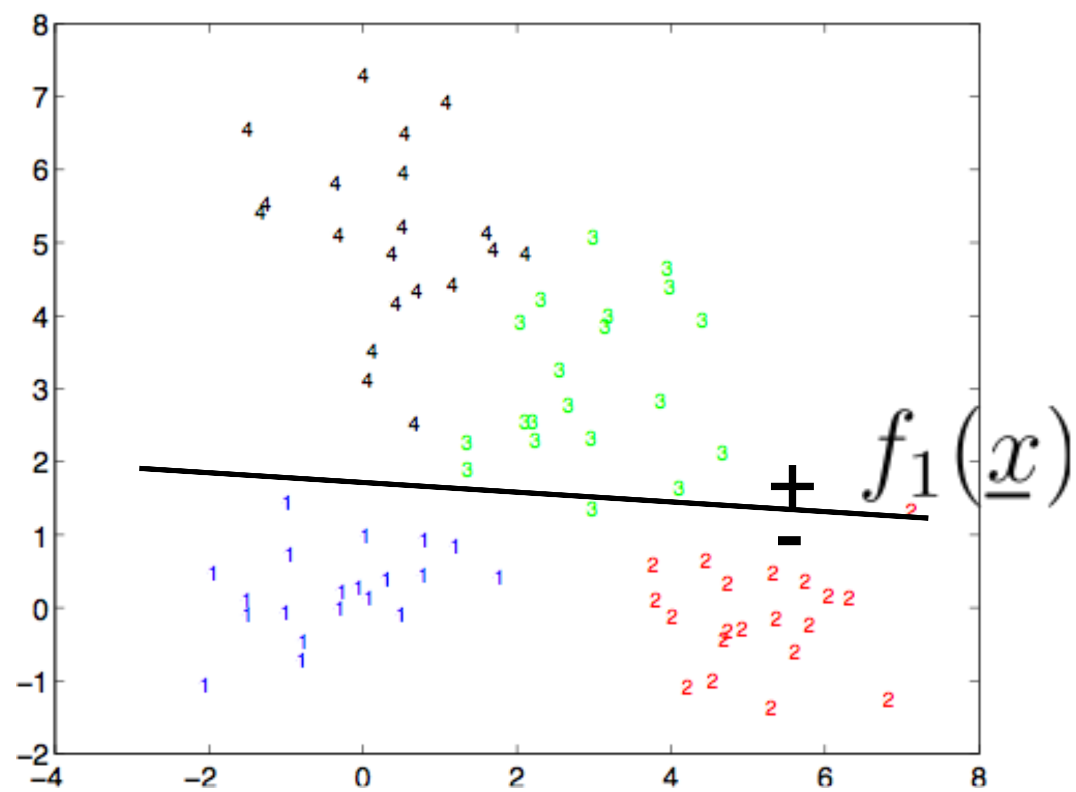
Today's topics

- Brief review
  - support vector machine with kernels
- One-class problems, anomaly detection
  - simple formulation, dual
  - removing outliers
- Multi-way classification
  - reducing multi-class to binary
  - margin based solution

Multi-way classification

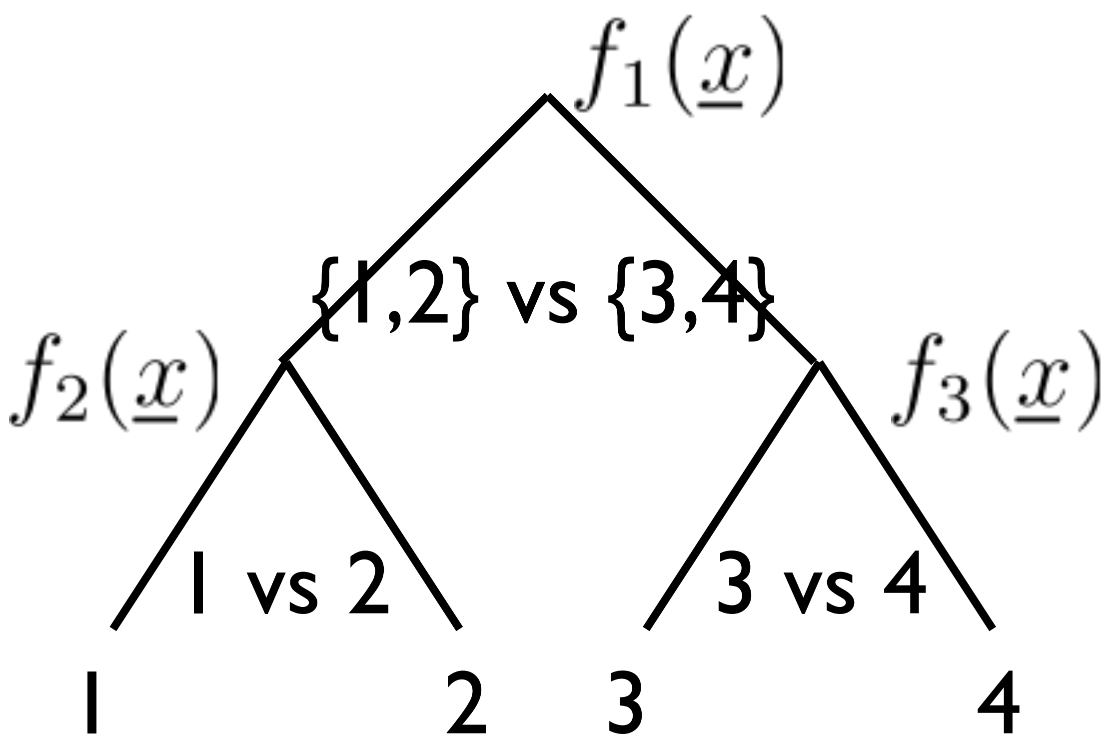
- Character recognition, face recognition, tumor identification, etc., are not binary classification problems
- We can, however, reduce multi-way classification problems to sets of binary classification problems

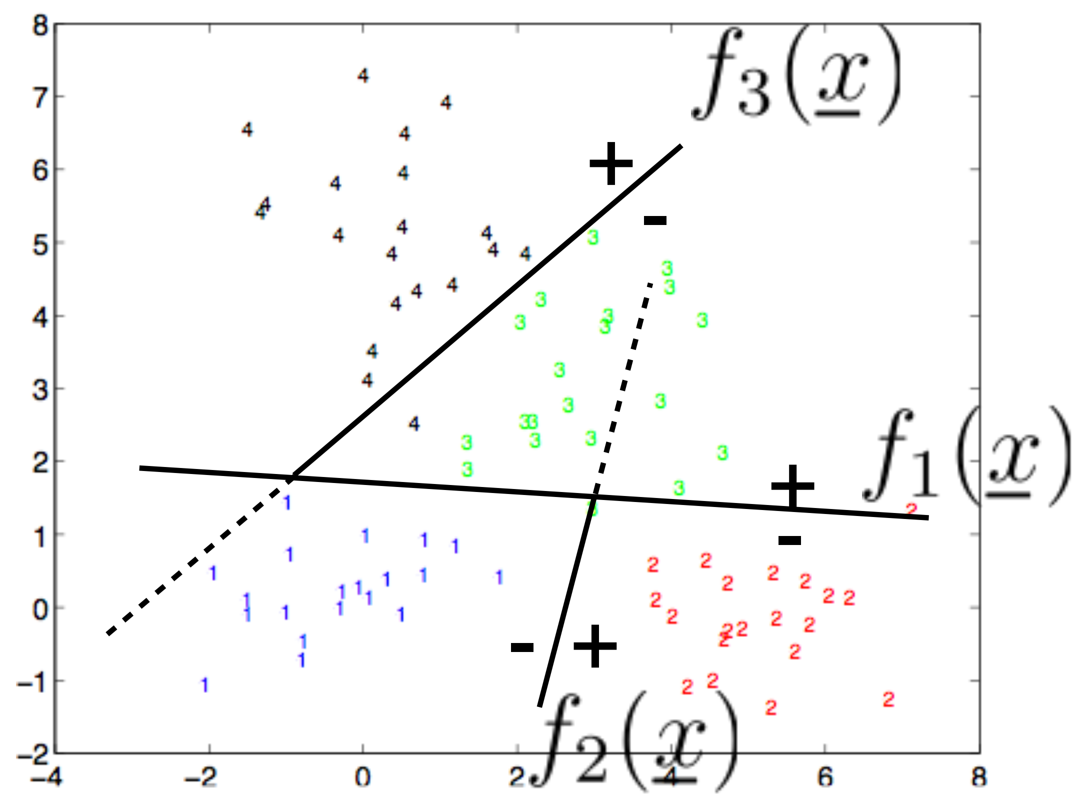




Multi-way classification

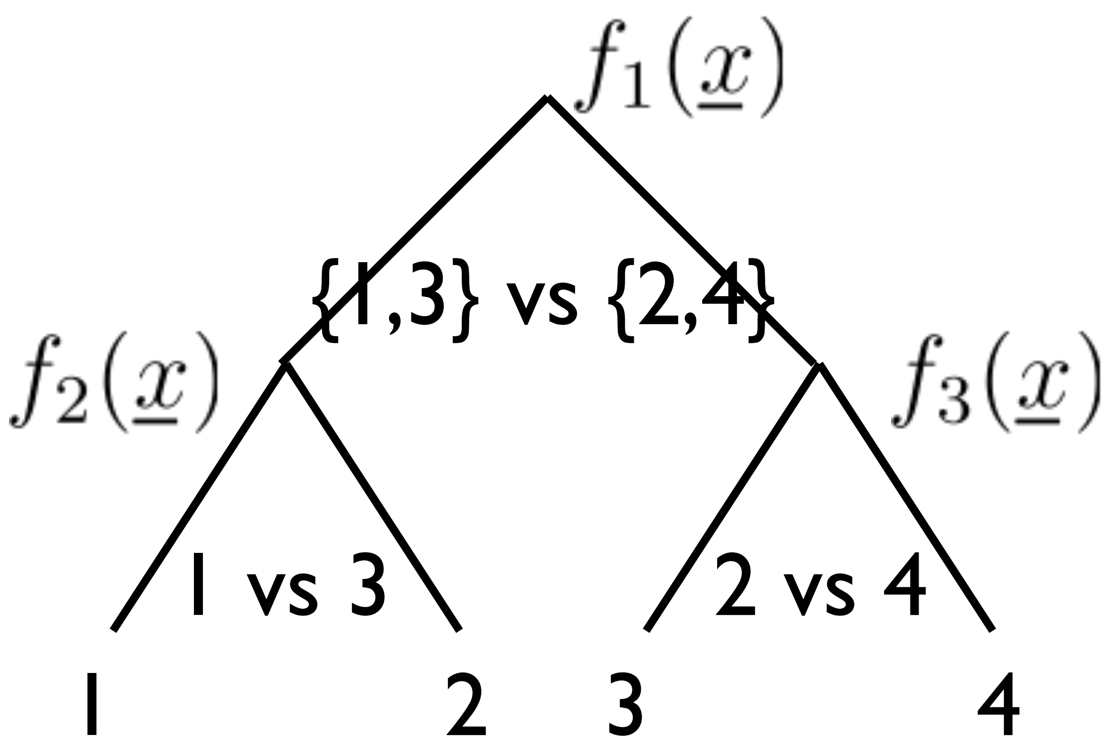
- Character recognition, face recognition, tumor identification, etc., are not binary classification problems
- We can, however, reduce multi-way classification problems to sets of binary classification problems

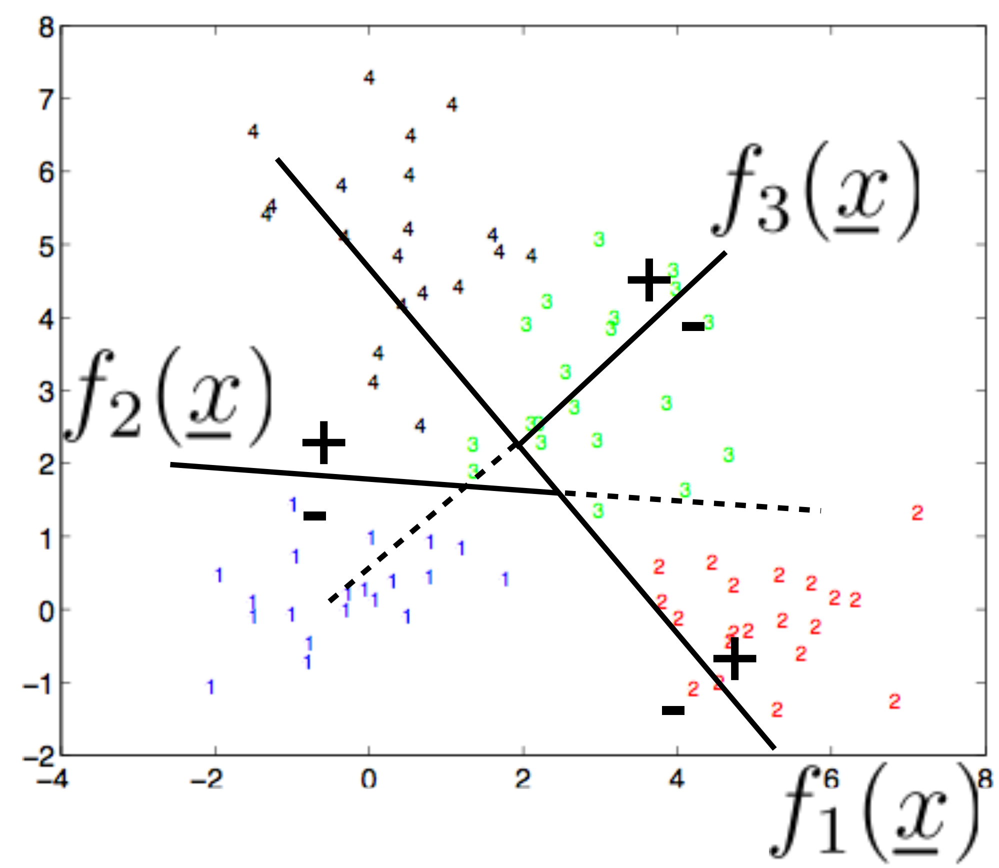




Reducing multi-class to binary

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





- Things to consider
  - accuracy we can achieve for each binary task
  - redundancy built into the partitioning scheme
  - 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

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

{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

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

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

classes

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

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

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

classes

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

4th classifier sees any training example not in class 4 labeled as -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

binary tasks

class pair 1, 2

$$
\text{classes} \quad
\begin{array}{c}
\\ 1 \\ 2 \\ 3 \\ 4
\end{array}
\begin{array}{c}
\begin{array}{cccccc} f_1(\underline{x}) & f_2(\underline{x}) & f_3(\underline{x}) & f_4(\underline{x}) & f_5(\underline{x}) & f_6(\underline{x}) \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 1 & 1 & 0 \\
0 & -1 & 0 & -1 & 0 & 1 \\
0 & 0 & -1 & 0 & -1 & -1
\end{bmatrix}
\end{array}
$$

all-pairs output code matrix R

- Imagine we send a test point $\underline{x}$ to the classifiers above

$f_1(\underline{x})$	$f_2(\underline{x})$	$f_3(\underline{x})$	$f_4(\underline{x})$	$f_5(\underline{x})$	$f_6(\underline{x})$
1	-1	1	-1	1	1

is point $\underline{x}$ class $y = 1$, $y = 2$, $y = 3$ or $y = 4$?

Reducing multi-class to binary

- We train several classifiers. For a test point, we output a *string* (multi way label). We then check which matrix row is closest to the string.

binary tasks

string for class 2

$$
\begin{array}{cc}
\text{classes} &
\begin{array}{c}
1 \\ 2 \\ 3 \\ 4
\end{array}
\left[\begin{array}{cccccc}
1 & 1 & 1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 1 & 1 & 0 \\
0 & -1 & 0 & -1 & 0 & 1 \\
0 & 0 & -1 & 0 & -1 & -1
\end{array}\right]
\end{array}
$$

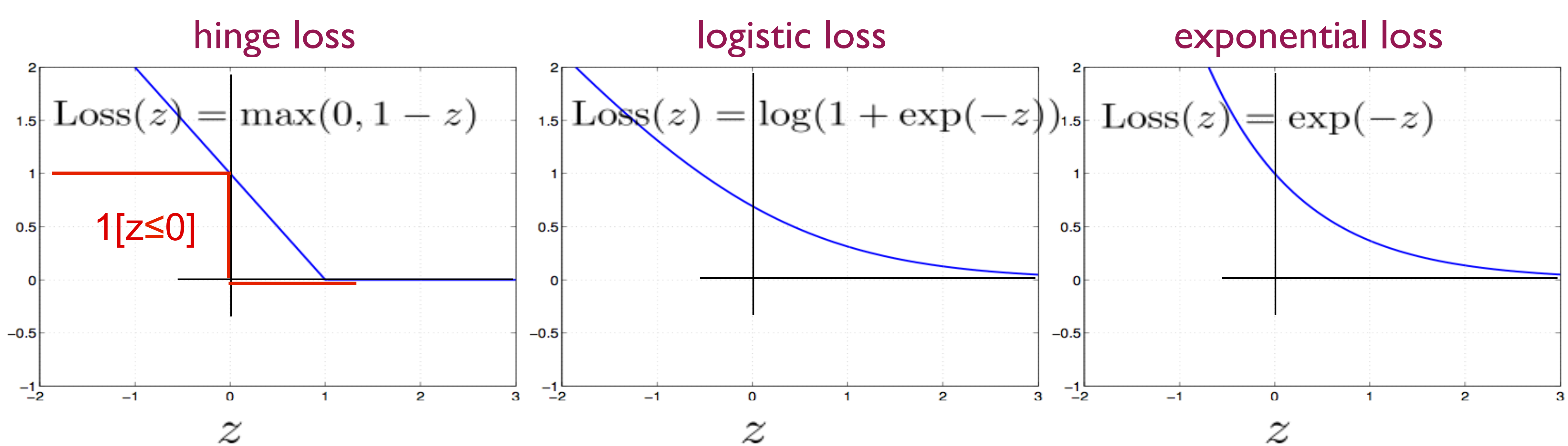
- Properties of good code matrices
  - "binary codes" (rows) should be well-separated: *if minimum Hamming distance between rows is H, we can make at most H/2 mistakes (good error correction)*
    - Which seems better: one-versus-all or all-pairs?
  - binary tasks (columns) should be easy to solve
    - Which seems better: one-versus-all or all-pairs?

Reducing multi-class to binary

- We train several classifiers. For a test point, we output a *string* (multi way label). We then check which matrix row is closest to the string.

binary tasks j

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



Reducing multi-class to binary

- We train several classifiers. For a test point, we output a *string* (multi way label). We then check which matrix row is closest to the string.

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

$$\hat{y} = \operatorname*{argmin}_y \sum_{j=1}^{m} \text{Loss}\Big(\overbrace{R(y,j)\,\underline{\hat{\theta}}_j \cdot \phi(\underline{x})}^{\text{agreement}} \Big)$$

target binary label for the jth classifier if the multi-class label is y

discriminant function value of the jth classifier in response to the new example

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 $H = \min_{y \neq y'} \Delta(y, y')$

m binary tasks

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

Output codes, error correction

- If the loss is the hinge loss $\mathrm{Loss}(z) = \max(0, 1 - z)$, then

$$\#\ \text{multi-class errors on the training set}$$
$$\leq \frac{1}{H} \sum_{t=1}^{n} \left[\sum_{j=1}^{m} \mathrm{Loss}\left(R(y_t, j)\, \underline{\hat{\theta}}_j \cdot \underline{\phi}(\underline{x}_t) \right) \right]$$

small if code words
are well-separated

small if each binary task
can be solved well

(See Theorem 1 in [2] if interested.)

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

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

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

- For new examples, we predict labels according to

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

(See [4] if interested.)