Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- Discriminative learning II
  - Multi-class classification
    - Using binary classifiers
    - Using multi-class SVM
  - Ranking
    - SVM Rank
  - Midterm Review (what are your questions?)

with much credit to T. Jaakkola

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

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

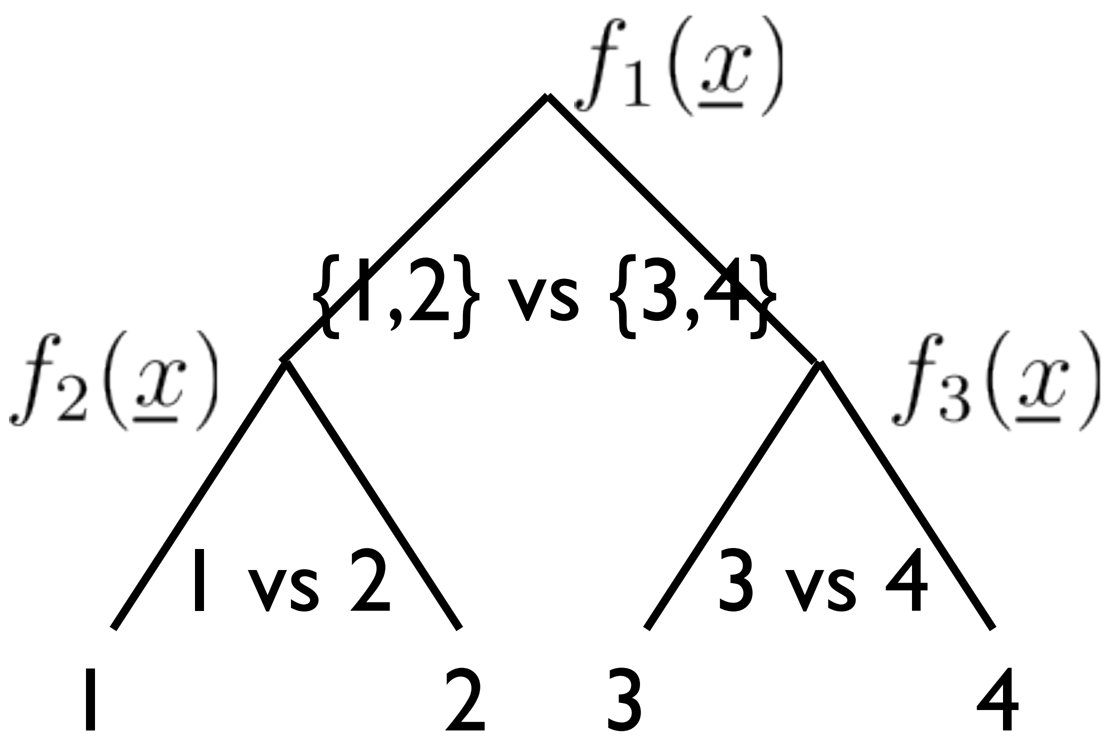
$$(\underline{\theta}_{y_i} \cdot \underline{\phi}(\underline{x}_i)) \geq (\underline{\theta}_{y'} \cdot \underline{\phi}(\underline{x}_i)) + 1, \ \forall y' \neq y_i, \ i = 1, \ldots, n$$

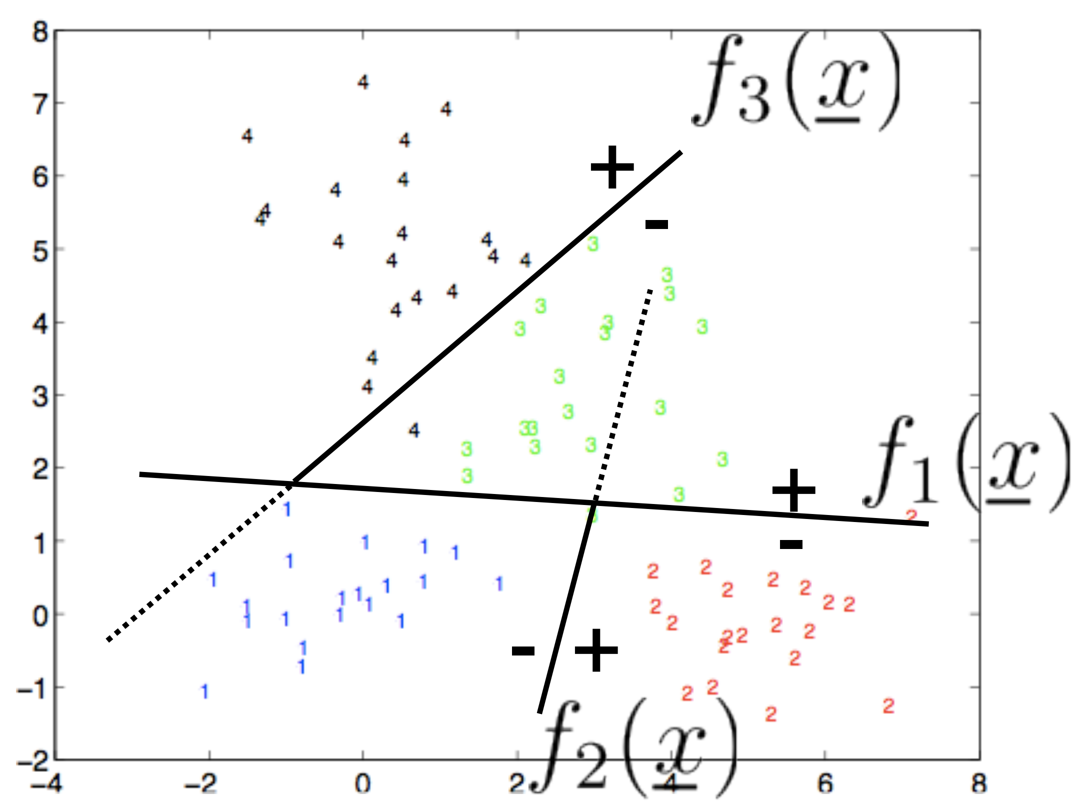
- For new examples, we predict labels according to

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

Multi-way classification

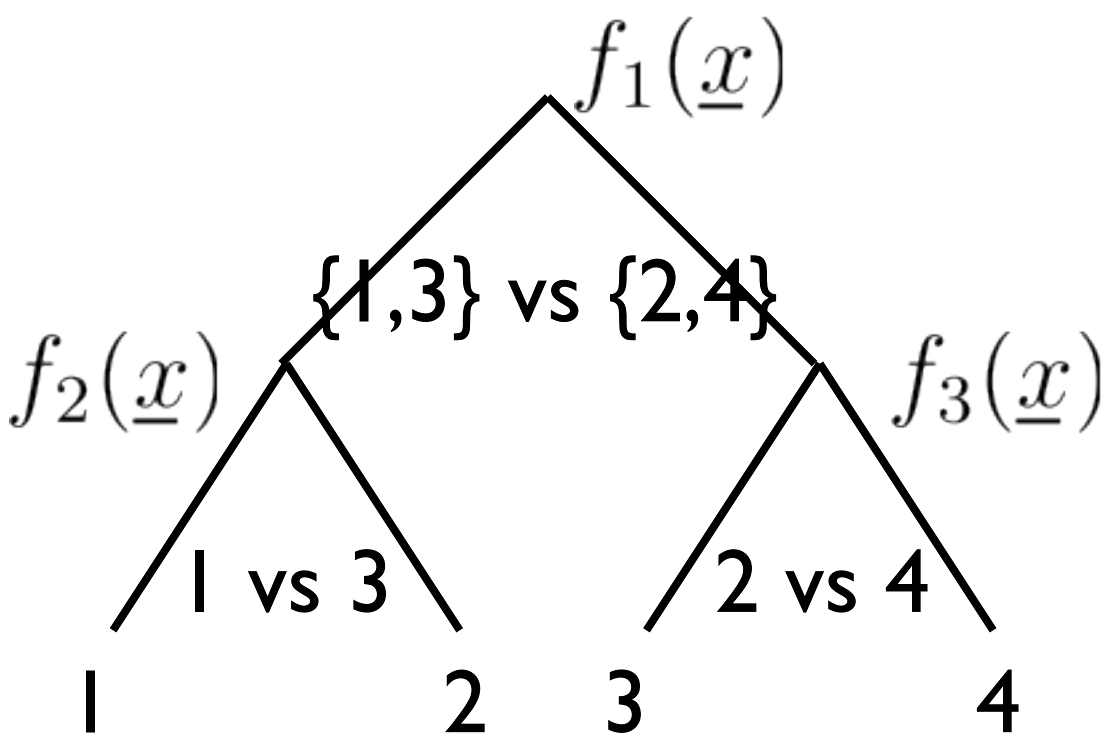
- Character recognition, face recognition, tumor identification, etc., are not binary classification problems
- We can, however, reduce multi-way classification problems to sets of binary classification problems

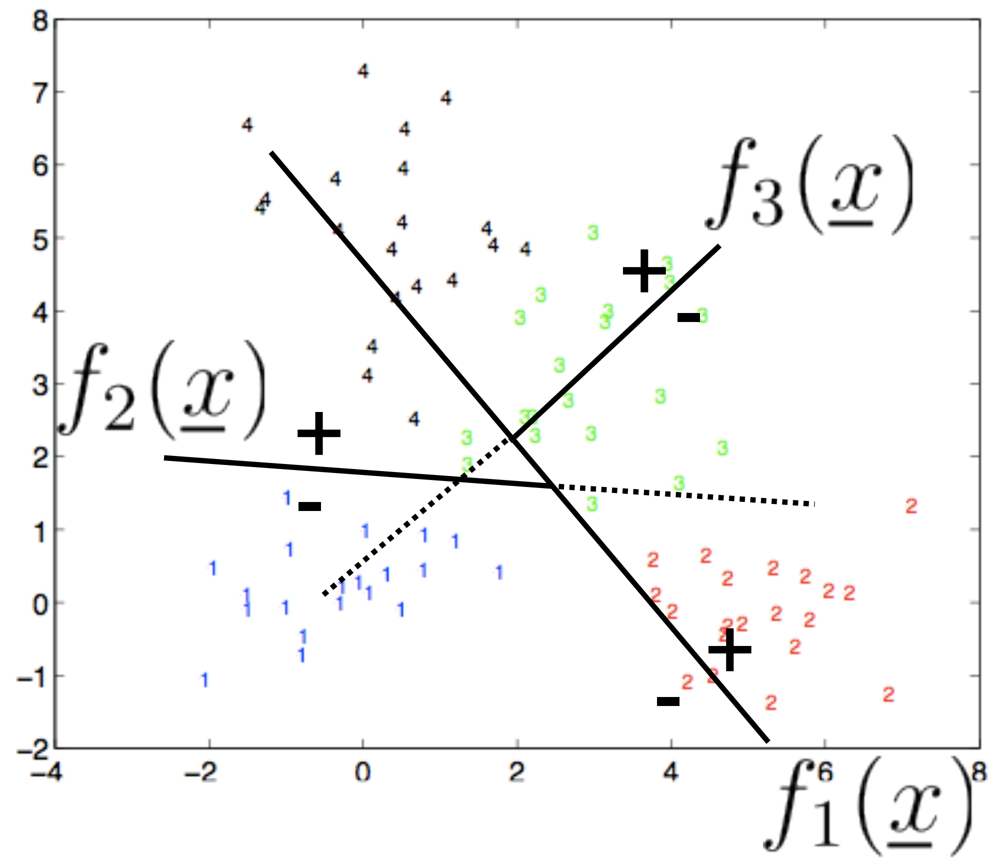




Reducing multi-class to binary

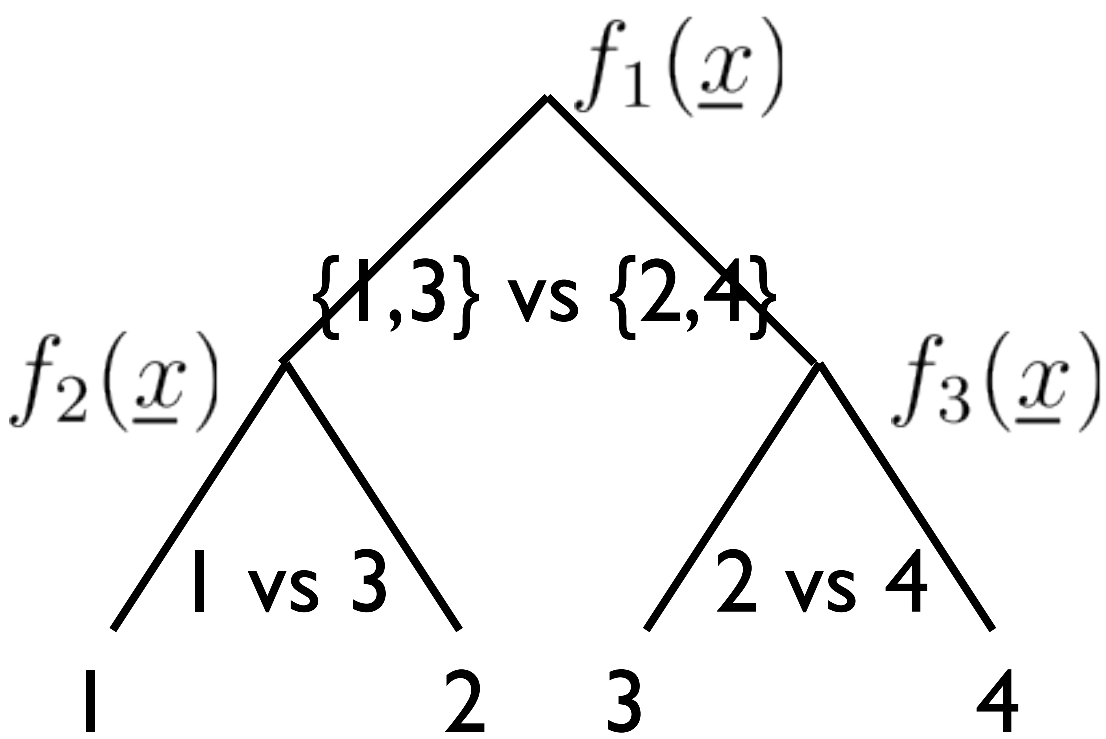
- How we partition the classes into binary problems matters a great deal

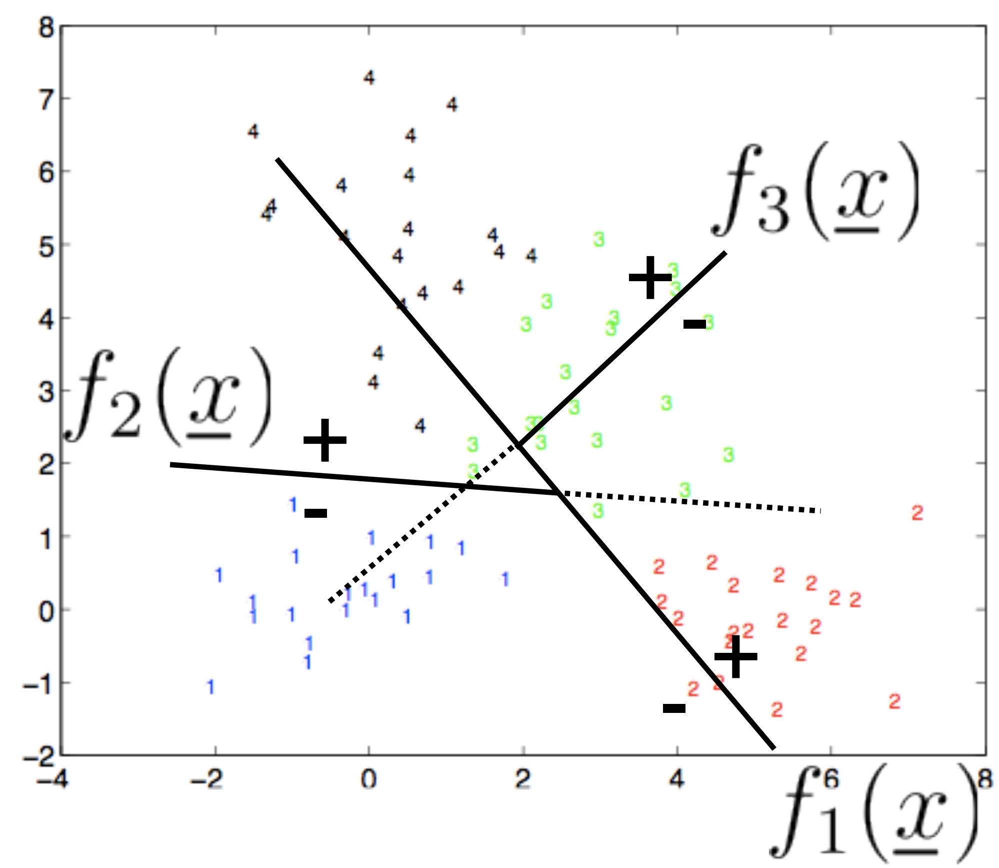




Reducing multi-class to binary

- 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

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

{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

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 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

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

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

the 3rd classifier sees any training example from class 4 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

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

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

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

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

m binary tasks

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

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)

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

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

Output codes, error correction

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

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

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

small if code words are well-separated

small if each binary task can be solved well

Rating / ordinal regression

- Rating products, movies, etc. using a few values (e.g., 1-5 stars) results in a partial ranking of the items
- Multi-class problems, where the labels have some ordering, can be addressed as ordinal regression or rating
- But what if the labels don't really matter so much as the ordering among objects?

Ranking

- What if the labels don't really matter as much as the ordering among objects?
- Many rating / classification problems are better viewed as ranking problems
  - suggest movies in the order of user interest in them
  - rank websites to display in response to a query (a.k.a. Websearch)
  - suggest genes relevant to a particular disease condition, etc.
- By casting the learning problem as a ranking problem we can also incorporate other types of data / feedback
  - e.g., click through data from users

Ranking example

- We would like to rank n websites (find top sites to display) in response to a few query words.

$$x = \text{context (set of query words)}$$
$$y = \text{website}$$

$$(x_1, y_2)$$
$$(x_1, y_{10})$$
$$(x_1, y_3)$$

$$\cdots$$
$$(x_1, y_n)$$

Ranking example

- We would like to rank n websites (find top sites to display) in response to a few query words

$$x = \text{context (set of query words)}$$
$$y = \text{website}$$

$$(x_1, y_2)$$
$$(x_1, y_{10})$$
$$(x_1, y_3) \quad \mathbf{x}$$

$$\cdots$$
$$(x_1, y_n)$$

- The available data contain user selections (clicks) of websites out of those displayed to them

From selections to preferences

- We can interpret a user click as a statement that they prefer the selected link over others displayed in the context of the query

$$(x_1, y_2)$$
$$(x_1, y_{10})$$
$$(x_1, y_3) \quad \mathbf{x}$$

$$\cdots$$
$$(x_1, y_n)$$

$$\Downarrow$$

$$(x_1, y_3) > \{(x_1, y_{10}), (x_1, y_2)\}$$

Ranking function

- Our goal is to estimate a ranking function over pairs f(x,y) such that its values are consistent with the observed preferences.

$$(x_2, y_7) > \{(x_2, y_2), (x_2, y_1)\}$$

$$\Rightarrow \quad f(x_2, x_7) > f(x_2, y_2), \quad f(x_2, x_7) > f(x_2, y_1)$$

- We can parameterize this function in terms of feature vectors computed on each (context, website) pair

$$f(x, y; \underline{\theta}) = \underline{\theta} \cdot \underline{\phi}(x, y)$$

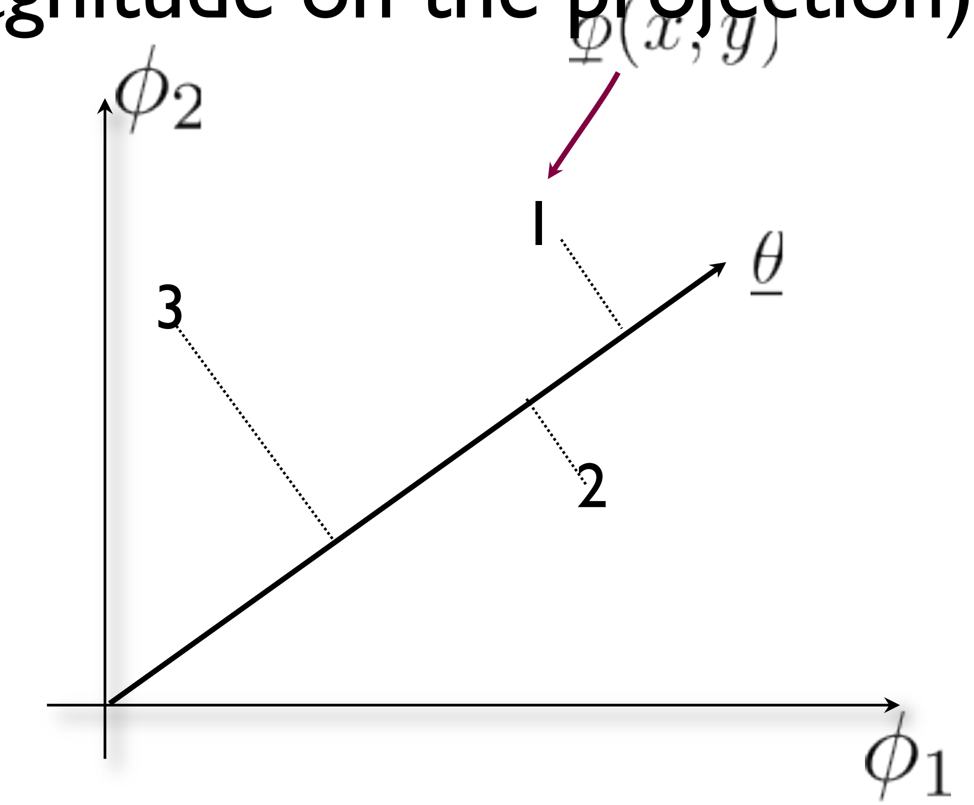
where the features could be, e.g.,

$$\phi_w(x, y) = \left\{ \begin{array}{ll} 1, & \text{if word } w \text{ appears in } x \text{ and } y \\ 0, & \text{otherwise} \end{array} \right\}$$

$$\text{for all } w \in \mathcal{W}$$

Ranking function

- The ranking function gives rise to a total ordering of the pairs via projection to the parameter vector (ranking is the ordering/magnitude on the projection).



$$f(x, y; \underline{\theta}) = \underline{\theta} \cdot \underline{\phi}(x, y)$$

SVM rank

- Training set: a set of order relations between pairs

$$D = \{ \{(x_i, y_j) > (x_k, y_l)\} \}$$

- An SVM-style algorithm for finding a consistent ranking function

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

$$\underline{\theta} \cdot \underline{\phi}(x_i, y_j) \geq \underline{\theta} \cdot \underline{\phi}(x_k, y_l) + 1,$$

$$\forall \{(x_i, y_j) > (x_k, y_l)\} \text{ in } D$$

SVM rank

- A training set of order relations between pairs

$$D = \{\, \{(x_i, y_j) > (x_k, y_l)\} \,\}$$

- An SVM style algorithm for finding a consistent ranking function

$$\text{minimize} \quad \frac{1}{2}||\underline{\theta}||^2 + C \sum_{ij;kl} \xi_{ij;kl} \quad \text{subject to}$$

$$\underline{\theta} \cdot \underline{\phi}(x_i, y_j) \geq \underline{\theta} \cdot \underline{\phi}(x_k, y_l) + 1 - \xi_{ij;kl}, \quad \xi_{ij;kl} \geq 0$$

$$\forall \, \{(x_i, y_j) > (x_k, y_l)\} \text{ in } D$$

- It is important to appropriately weight or choose which constraints to include

SVM rank

- A training set of preference relations between pairs

$$D = \{ \{(x_i, y_j) > (x_k, y_l)\} \}$$

- We use c to index the preference relations

$$c \in D, \quad \text{e.g.,} \quad c = \{(x_i, y_j) > (x_k, y_l)\}$$

- An SVM style algorithm for finding a consistent ranking function can then be written as

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

$$\underline{\theta} \cdot \delta\underline{\phi}(c) \geq 1, \quad \forall\, c \in D$$

$$\text{where} \quad \delta\underline{\phi}(c) = \underline{\phi}(x_i, y_j) - \underline{\phi}(x_k, y_l),$$

$$\text{if} \quad c = \{(x_i, y_j) > (x_k, y_l)\}$$

Note that this is formulated as a (binary) SVM over the difference vectors $\delta\underline{\phi}(c)$

SVM Rank

- If we choose to omit aυ-fraction of preference constraints, the dual problem is given by

$$\text{maximize} \quad -\frac{1}{2}\sum_{c,c'\in D} \alpha_c \alpha_{c'} \left[\delta\underline{\phi}(c)\cdot\delta\underline{\phi}(c')\right]$$

kernel over difference vectors

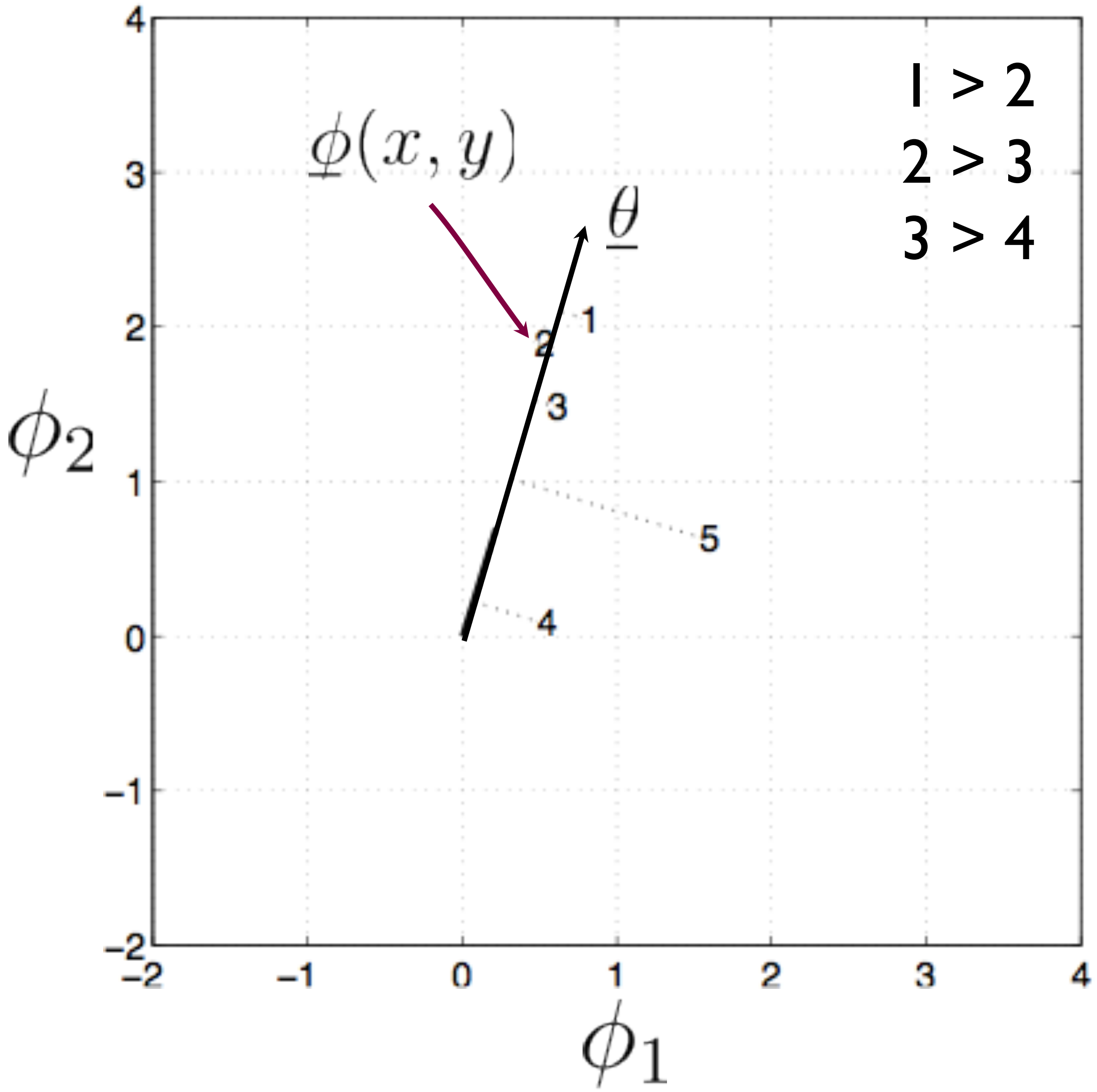
$$\text{subject to} \quad 0 \le \alpha_c \le \frac{1}{\nu|D|}, \quad \sum_{c\in D}\alpha_c = 1$$

sum over preference constraints in the training set

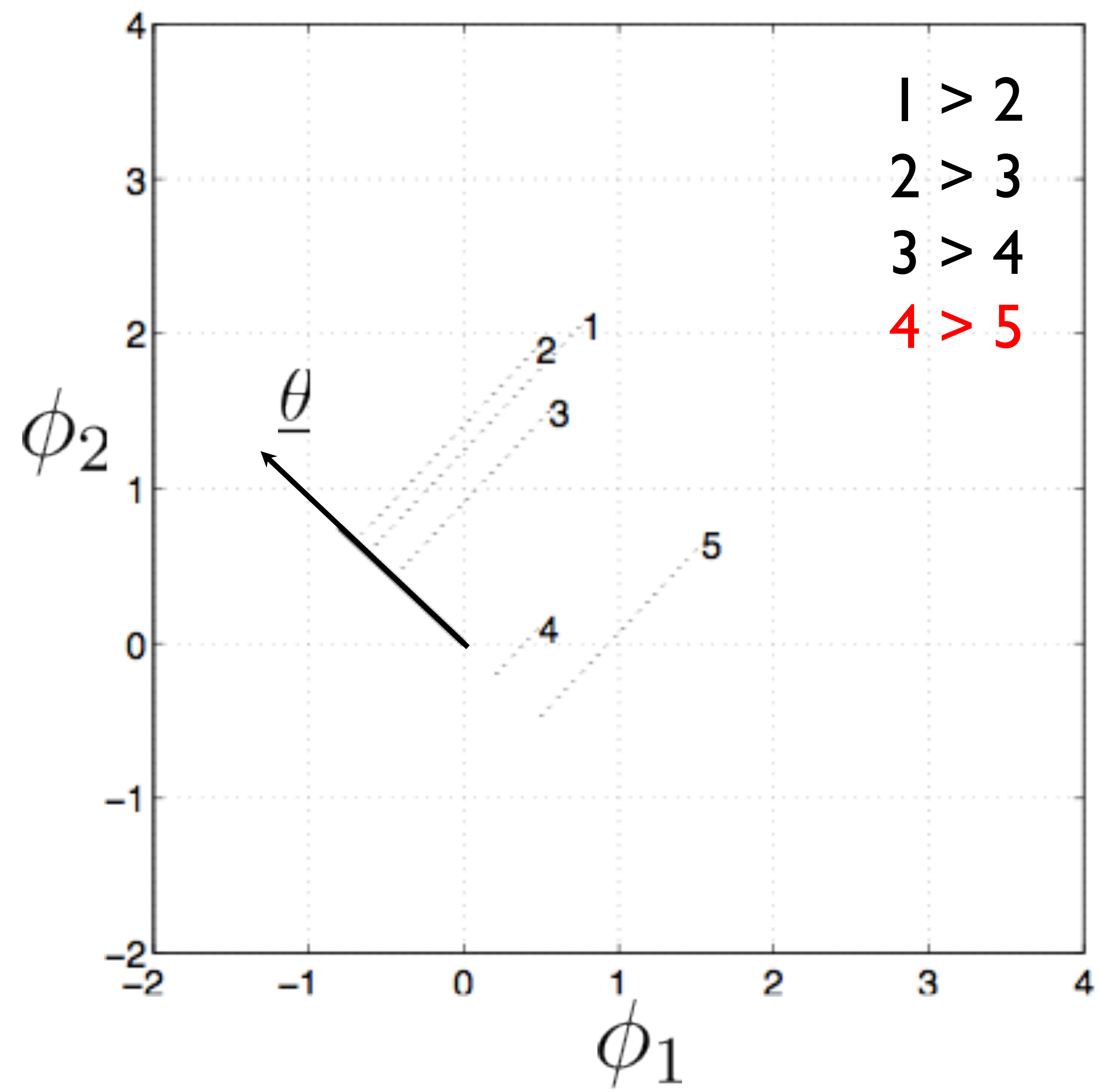
$$\underline{\theta}(\alpha^*) = \sum_{c\in D}\alpha_c^*\,\delta\underline{\phi}(c)$$

- The kernel over difference vectors $\delta\underline{\phi}(c)$ can be defined in terms of a kernel over the original feature vectors by expanding the difference vectors

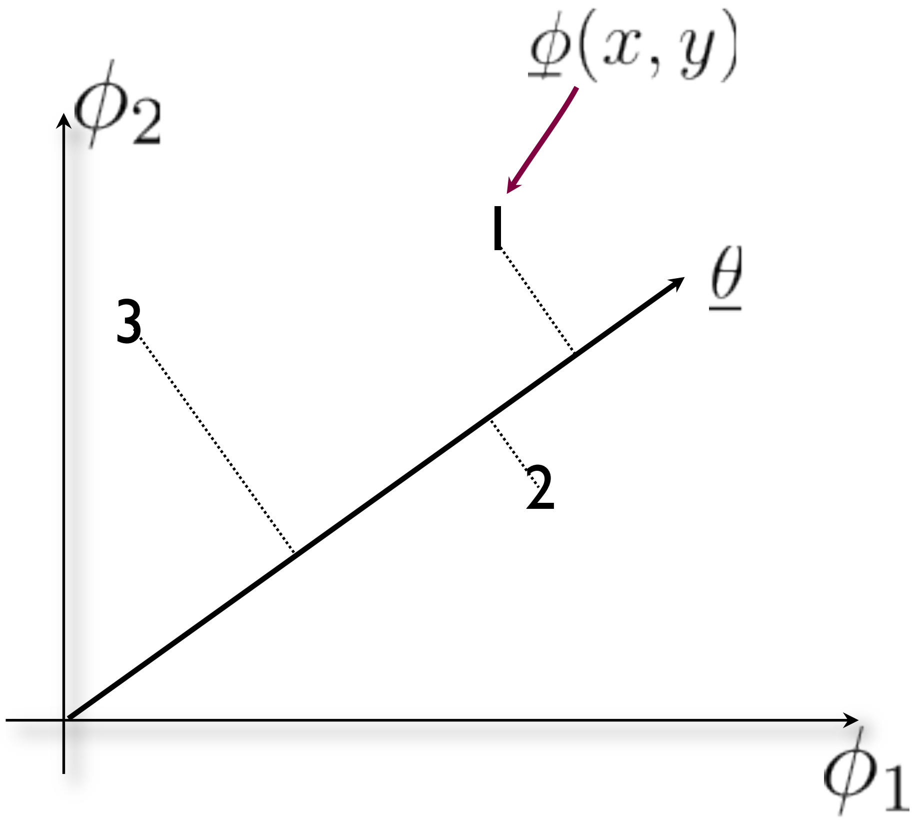
The effect of ranking constraints



adding a single constraint can have a large effect on the ranking solution

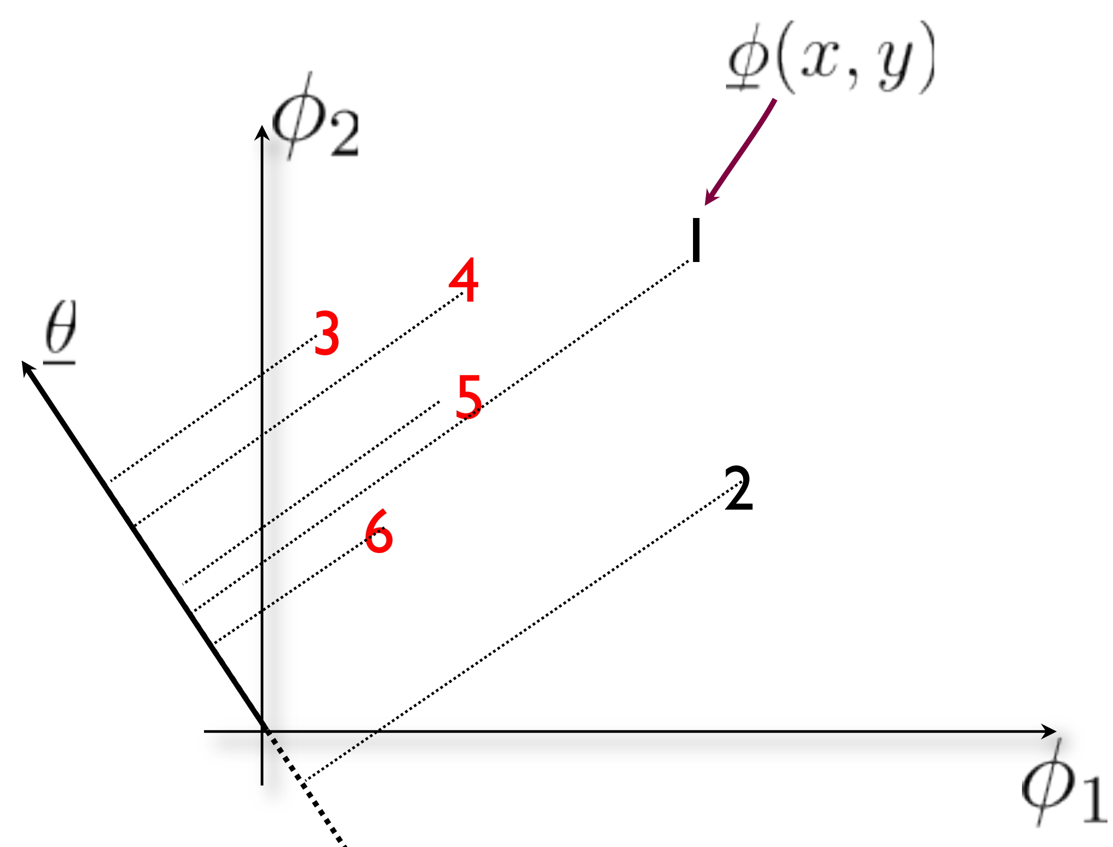


The effect of ranking constraints



1 > 2
1 > 3

good solution, no violated constraints



poor solution, only one violated constraint

1 > 2
1 > 3
3 > 4
4 > 5
5 > 6

Ranking postscript

- More recent advances include algorithms to put more weight on the ordering at the top of the ranked list
  - E.g. when doing a websearch, the user typically only looks at the first page of results
  - Even within the first page, the user typically looks at the top few results
- See, for example, publications of Cynthia Rudin (MIT)
  - E.g. the “P-norm push” ranking algorithm