Today

- Last time
  - SVMs with kernels (reading: Bishop Sec 7.1)
  - Multiclass classification (reading: Bishop, Sec 7.1.3)

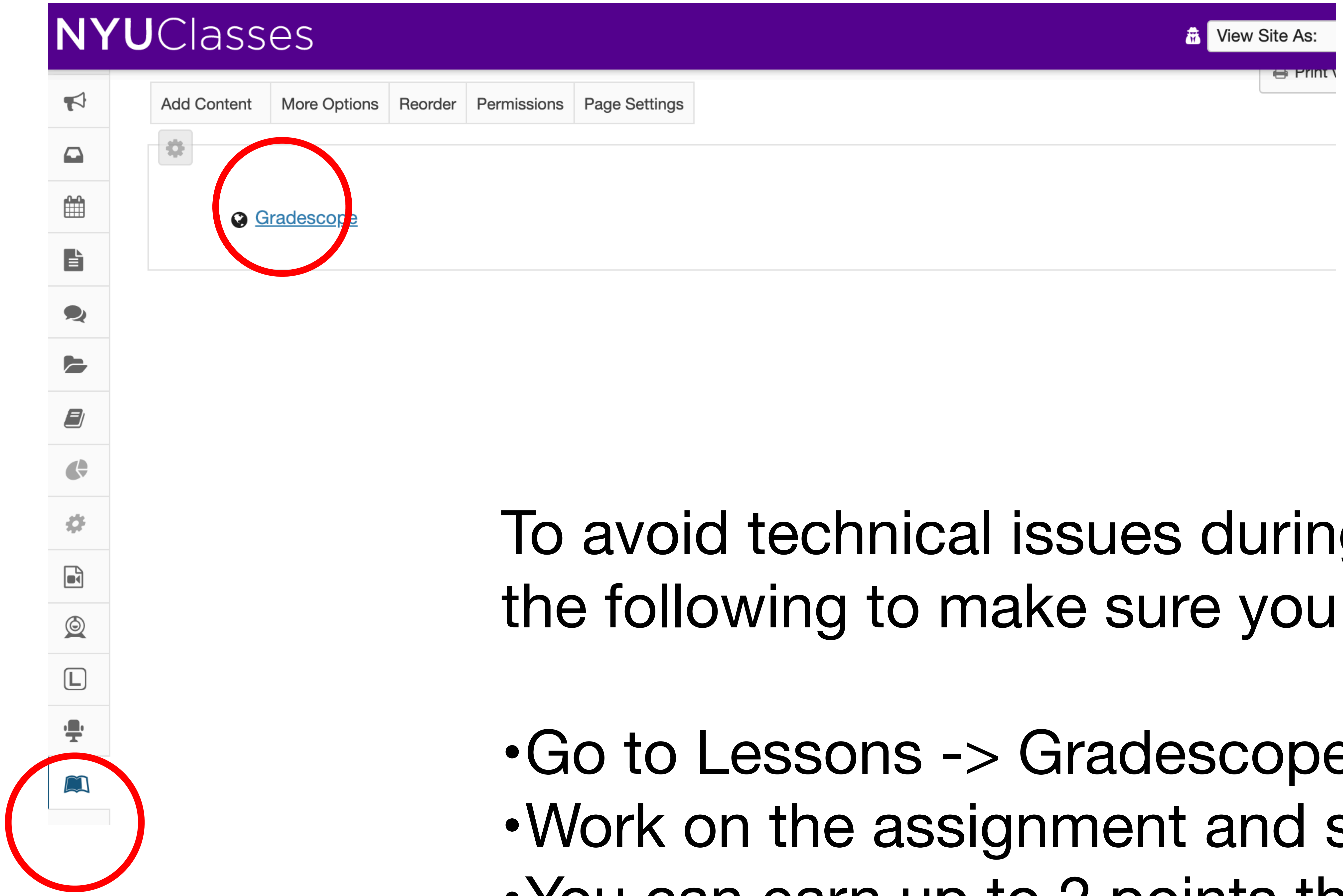
- Today
  - Optimization for SVM
  - Nearest neighbor classifiers (Bishop, Sec 2.5.2, Sec 6.3; Hastie, Sec 13)
  - Some learning theory (Ng, Lecture notes http://cs229.stanford.edu/notes/cs229-notes4.pdf)

- Announcements
  - HW 2 due on Wed, Oct 14
  - Lab on Wed, Oct 14; blended (know your seat number)
  - Midterm exam: Wed, Oct 21, 11.00am - 12.15pm (New York time)
    - Q&A on Mon, Oct 19 in lecture
    - Midterm online via Gradescope (more details to come)

How to prepare for the midterm exam

- Lecture (+)
  - Good for getting the “big picture” and a general understanding
  - Work through the derivations and try to understand each step
- Homeworks (++)
  - Learn how to implement concepts discussed in lecture
  - Theory questions useful to test your understanding (expect similar style of questions on the exam)
- Textbooks have many more exercises (+++)
  - Bishop, Pattern Recognition and Machine Learning
  - Hastie, The elements of statistical learning (data mining, inference, and prediction)
  - Murphy, Machine learning
- Make the most out of study groups, office hours, Q&A session, etc.

Gradescope



To avoid technical issues during the midterm exam, try the following to make sure you setup is working properly:

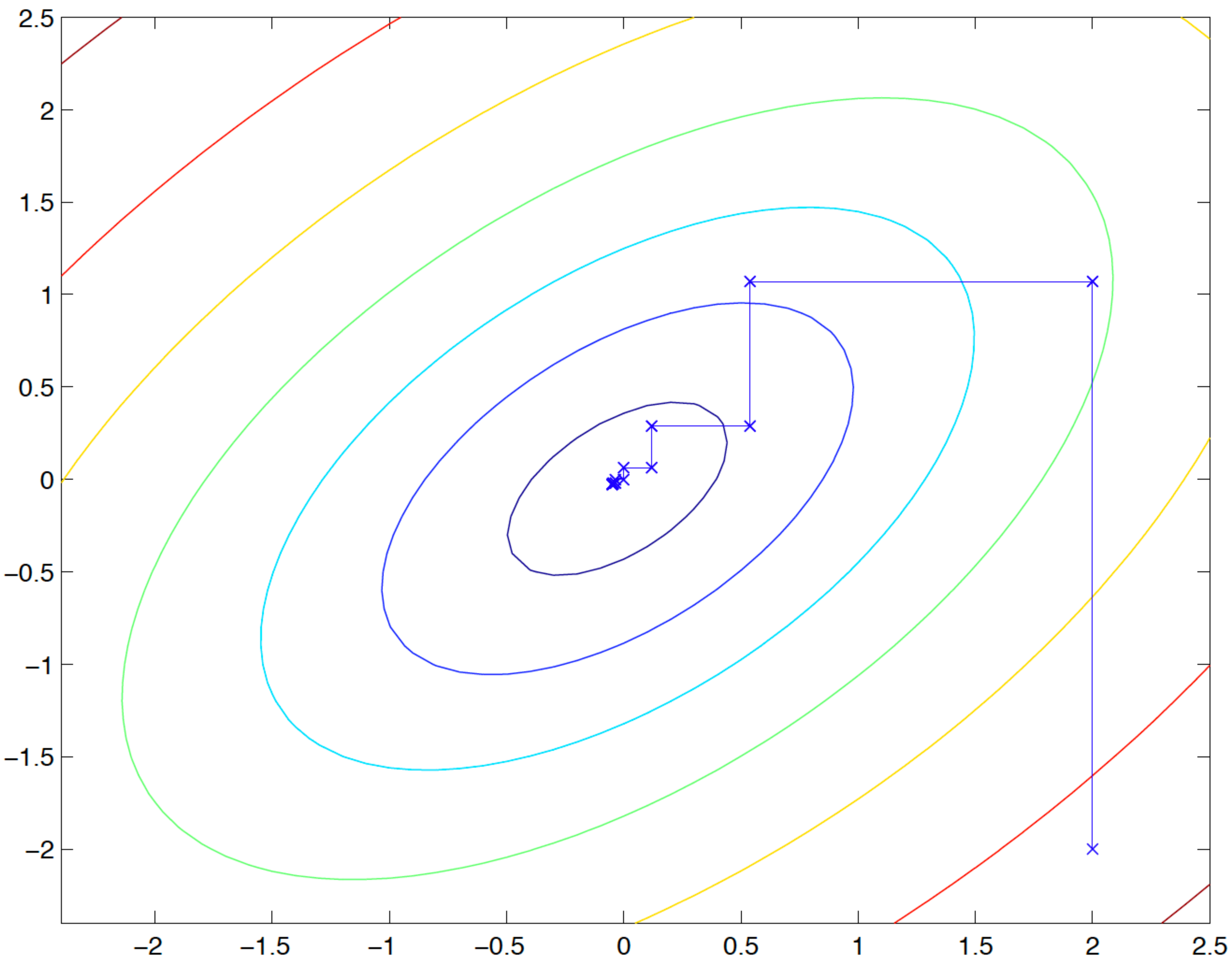
- Go to Lessons -> Gradescope -> TestYourSetup assignment
- Work on the assignment and submit
- You can earn up to 2 points that are counted towards the midterm

The assignment is open from today until Wed before class.

Algorithm for solving SVM problems

board

Coordinate descent



[Andrew Ng]

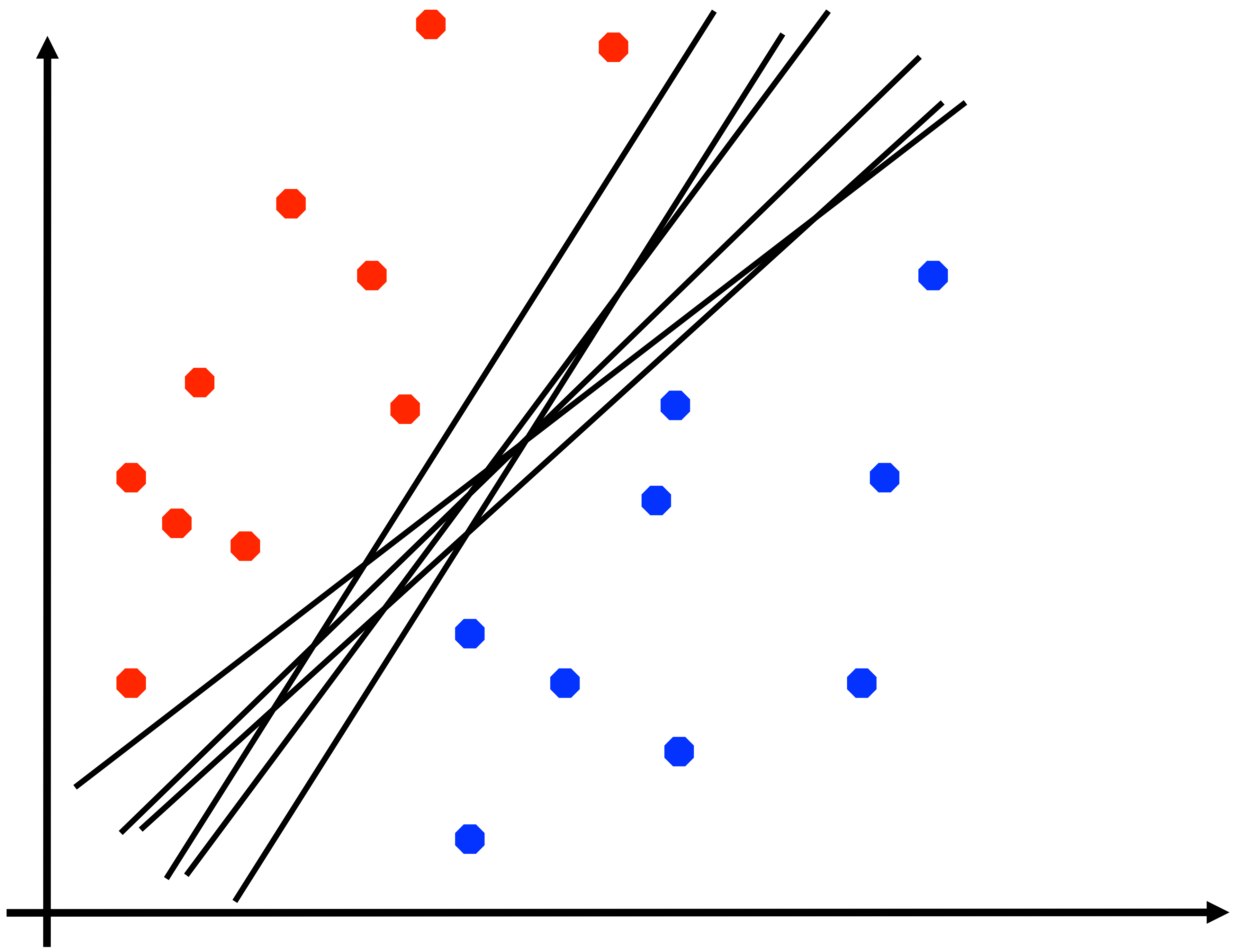
SMO algorithm

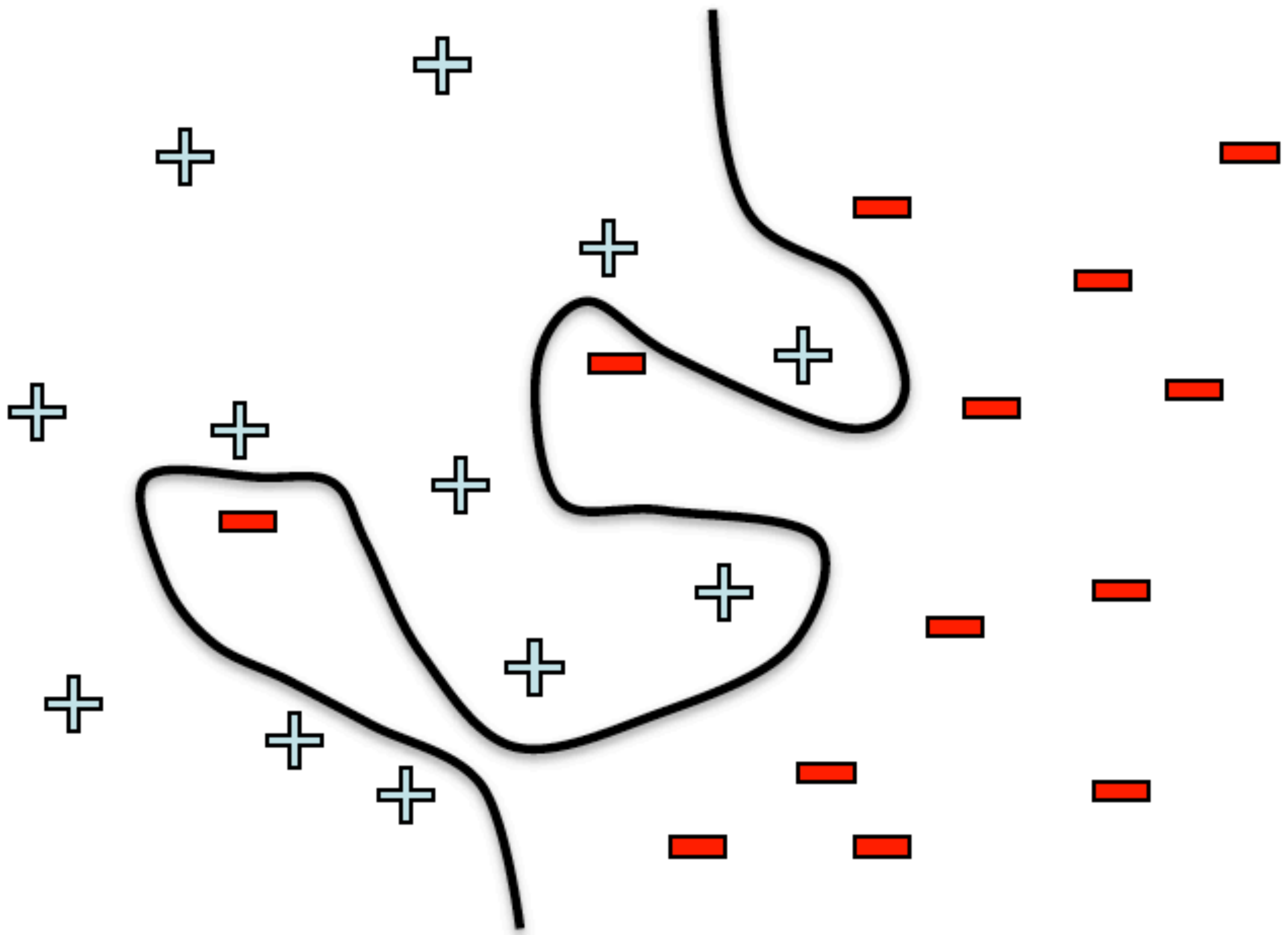
board

Nonlinear classifiers

Material based on David Sontag’s slides, Andrew Ng’s notes, and Cho’s notes

Recall: From linear to nonlinear decision boundary





[David Sontag]

Recall: Feature map

- Define a feature map $\phi : \mathbb{R}^n \to \mathbb{R}^k, k \in \mathbb{N} \cup \{\infty\}$
- Polynomial regression with degree 3

$$\phi(x) = \begin{bmatrix} x^3 \\ x^2 \\ x \\ 1 \end{bmatrix}$$

- Let's apply (linear) SVM to $\phi(x)$ instead of x
- Potential problem?
    - The dimension k potentially very (infinite) high
    - Goal: Avoid operating in $\mathbb{R}^k$

Example: XOR problem

0	0	1
0	1	0
1	0	0
1	1	1

Example: XOR problem (cont'd)

- This data set is ***not*** linearly separable

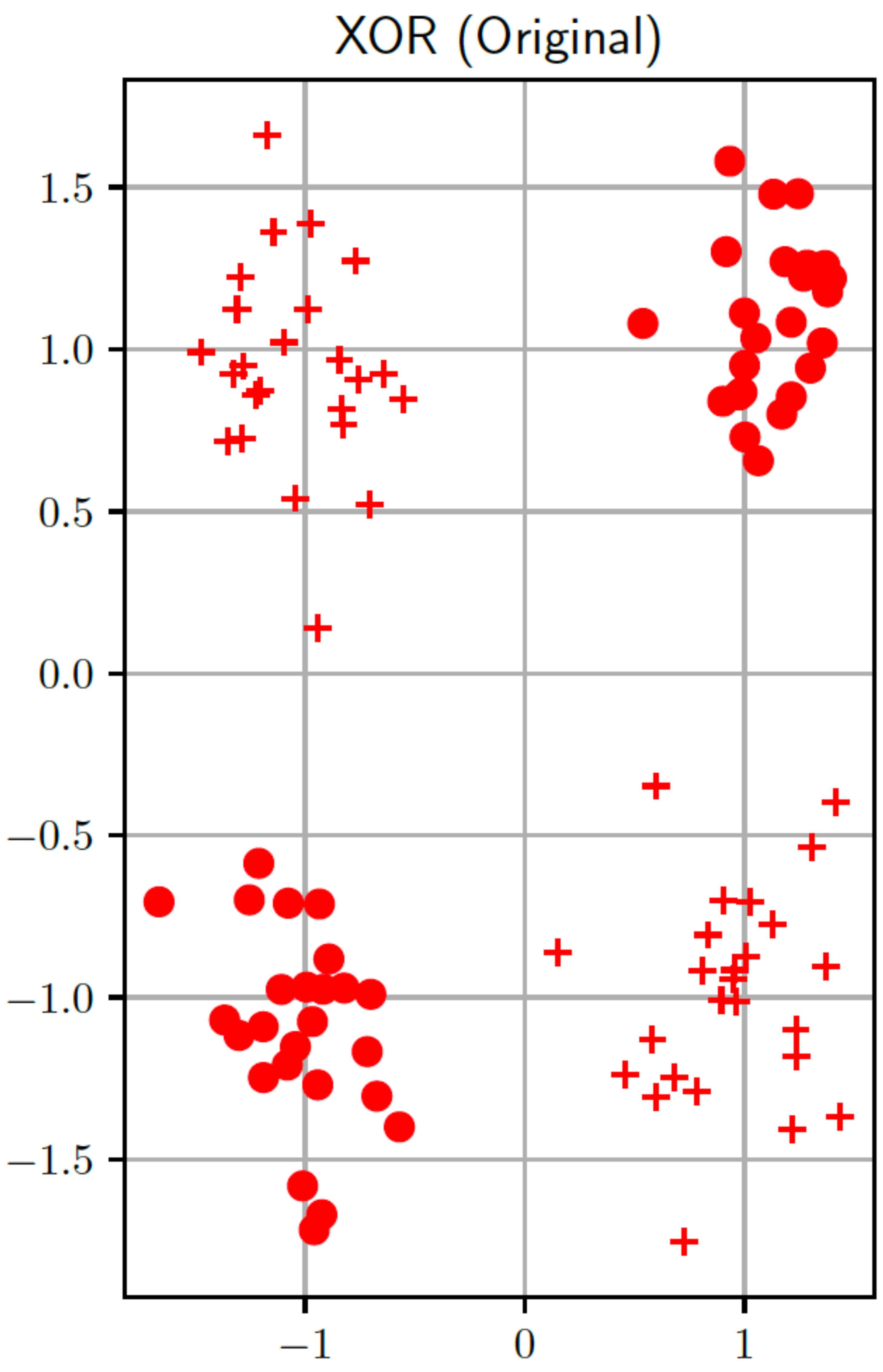
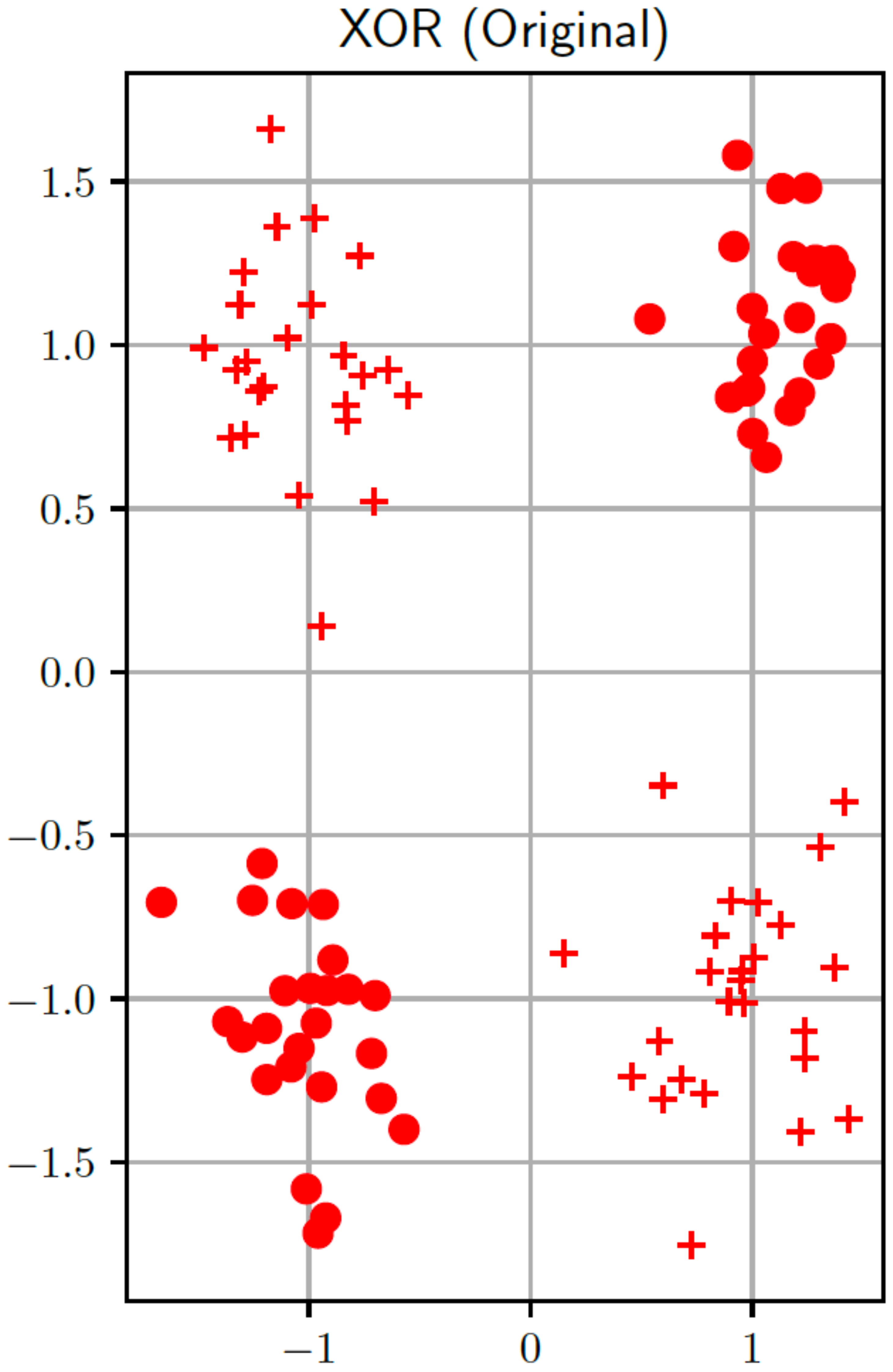


Figure: Cho

Feature mapping for XOR problem



Rotate with

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\boldsymbol{x}^r = R(\pi/4)\boldsymbol{x}$$

Still not linearly separable

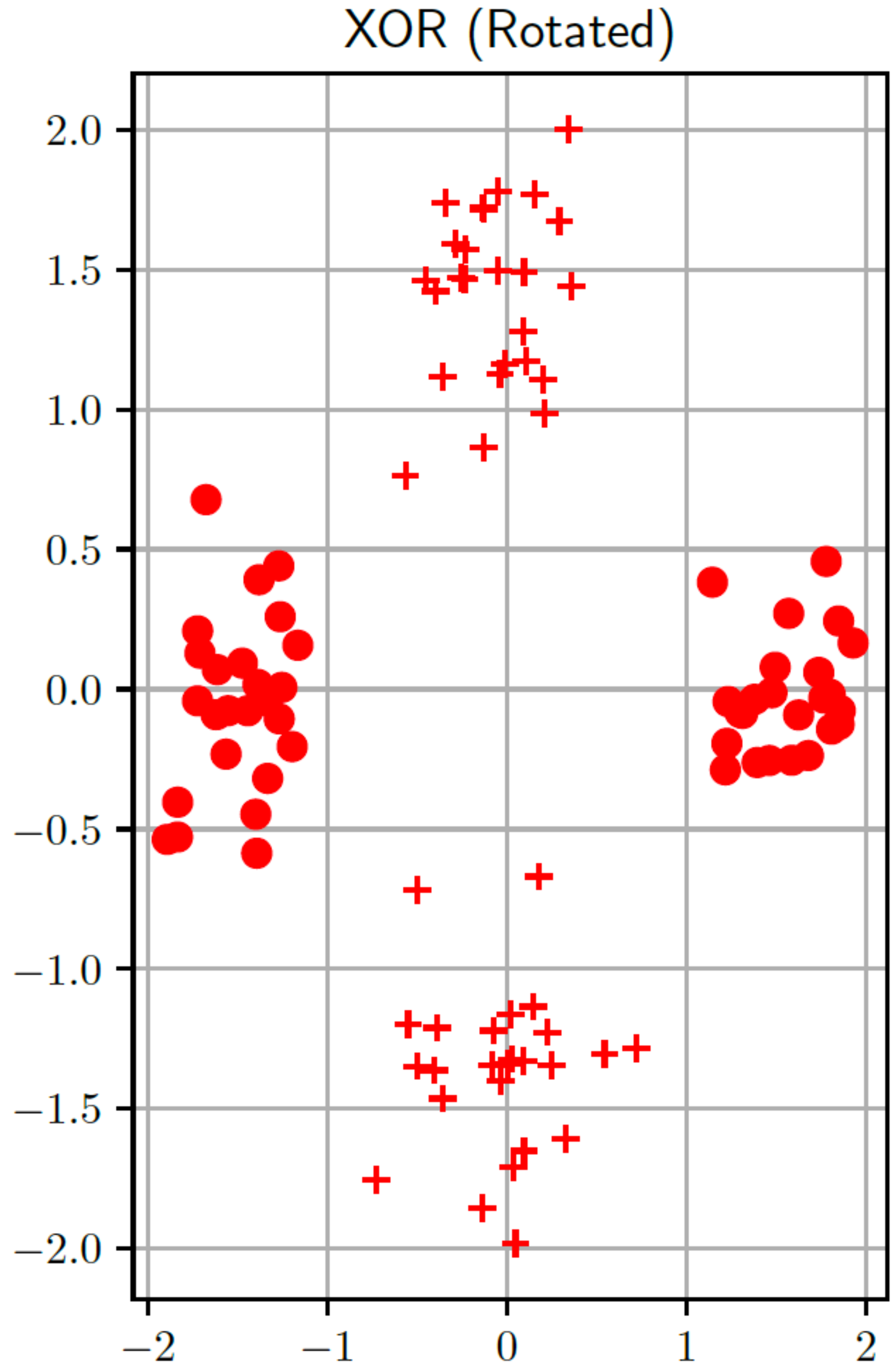
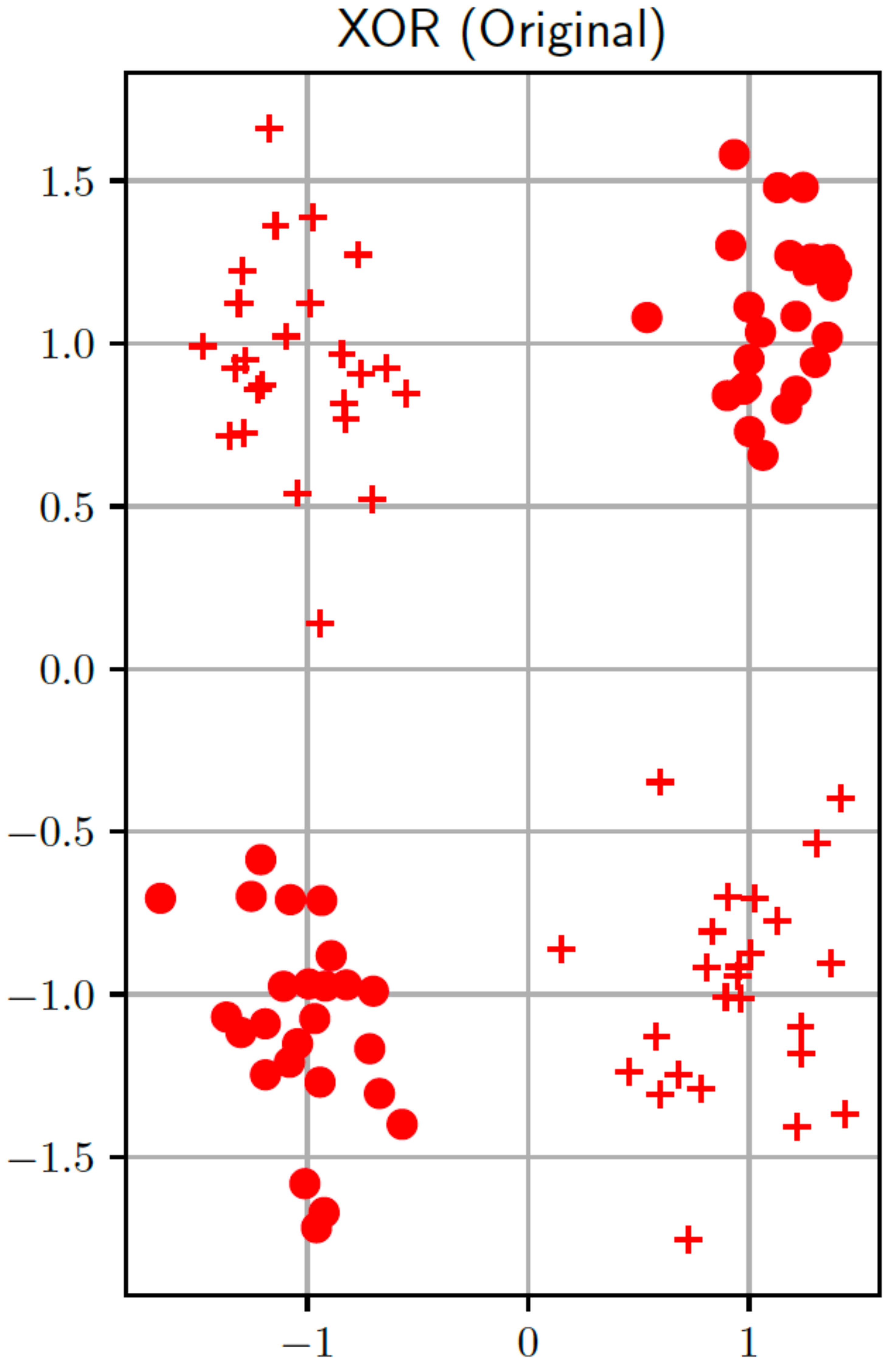


Figure: Cho

Feature mapping for XOR problem (cont'd)



Take absolute value of each rotated vector

$$\boldsymbol{x}^{ra} = |R(\pi/4)\boldsymbol{x}| = \phi(\boldsymbol{x})$$

Now it is linearly separable

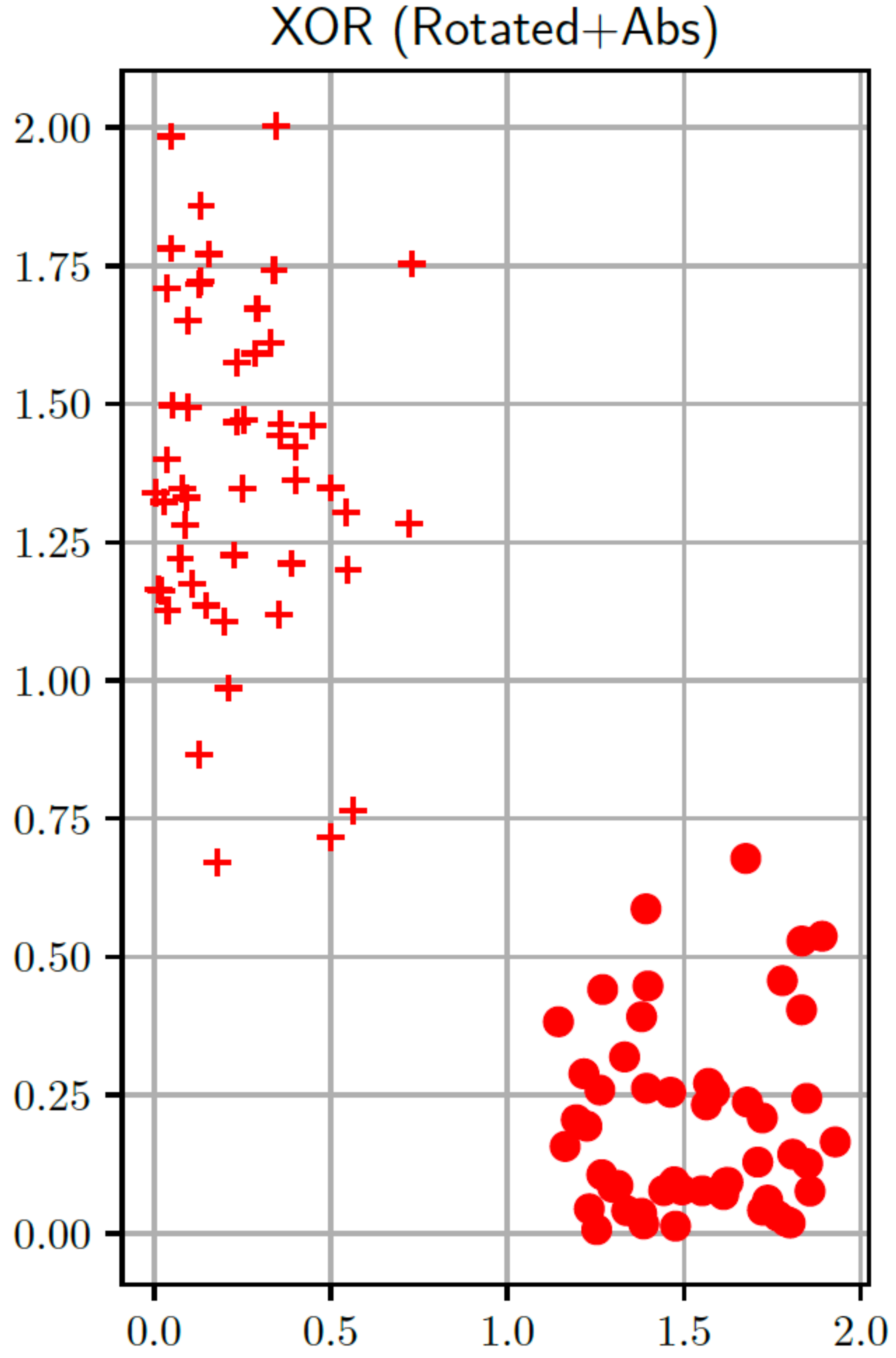


Figure: Cho

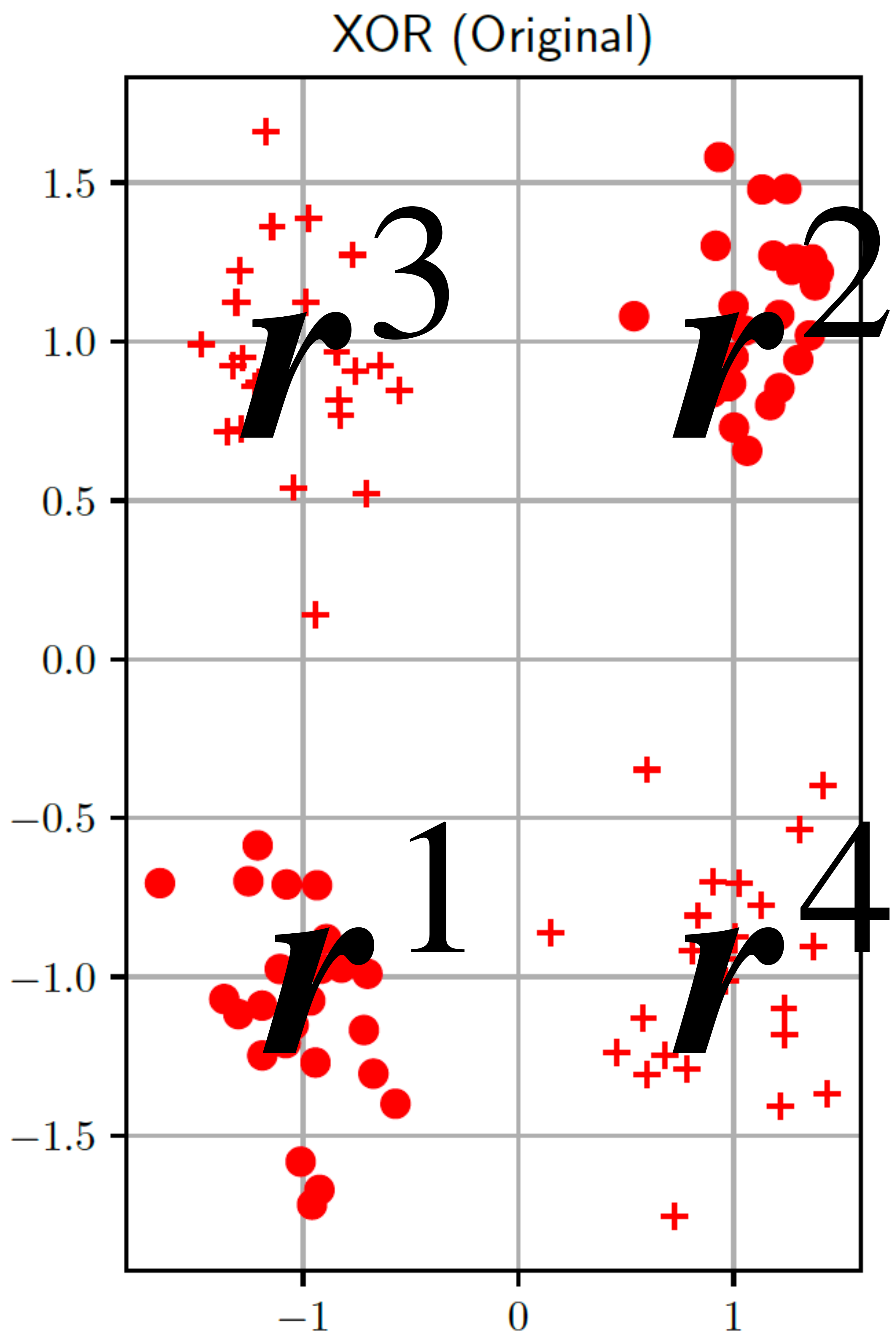
Another transformation for XOR problem

- Select 4 ``basis vectors”
$r^1 = [-1, -1]^T, r^2 = [1,1]^T, r^3 = [-1,1]^T, r^4 = [1, -1]^T$
- Define the feature mapping

$$\phi(\boldsymbol{x}) = \begin{bmatrix} \phi_1(\boldsymbol{x}) \\ \phi_2(\boldsymbol{x}) \\ \phi_3(\boldsymbol{x}) \\ \phi_4(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \exp(-\|\boldsymbol{x} - \boldsymbol{r}^1\|^2) \\ \exp(-\|\boldsymbol{x} - \boldsymbol{r}^2\|^2) \\ \exp(-\|\boldsymbol{x} - \boldsymbol{r}^3\|^2) \\ \exp(-\|\boldsymbol{x} - \boldsymbol{r}^4\|^2) \end{bmatrix}$$

- The i-th component is inverse proportional to the distance between $\boldsymbol{x}$ and vector $\boldsymbol{r}^i$

Closeness



- Close to r^1, r^2 should be classified as positive (circles)
  - Input vector in positive class are close to either r^1 or r^2

- Close to r^3, r^4 should be classified as negative (plus)
  - Input vector in positive class are close to either r^3 or r^4

Figure: Cho

Weight vector for linear classifier

- Class labels $y \in \{-1,1\}$
- Parametrize with θ

$$h_\theta(x) = g(w^T x)$$

with

$$g(z) = \begin{cases} 1\,, & z \geq 0 \\ -1\,, & z < 0 \end{cases}$$

- Weight vector $\theta = [1,1,-1,-1]^T$ perfectly solves XOR problem for transformed data $\phi(x)$
  - Proof: Board
- Note that $\theta = [y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}]$ is the vector containing the class label -1 or 1 of the basis vectors

Generalize this concept of constructing feature map

- Have K basis vectors with their labels

$$\{(\boldsymbol{r}^1, y^{(1)}), \ldots, (\boldsymbol{r}^K, y^{(K)})\}$$

- Each input is transformed as

$$\phi(\boldsymbol{x}) = \begin{bmatrix} \exp(-\|\boldsymbol{x} - \boldsymbol{r}^1\|^2) \\ \vdots \\ \exp(-\|\boldsymbol{x} - \boldsymbol{r}^K\|^2) \end{bmatrix}$$

- With weight vector

$$\boldsymbol{\theta} = [y^{(1)}, \ldots, y^{(K)}]^T$$

Nearest neighbor classifier

- Have N training points (= basis vectors) with their labels
$$\{(\boldsymbol{x}^1, y^{(1)}), \ldots, (\boldsymbol{x}^K, y^{(K)})\}$$
- Each input is transformed as
$$\phi(\boldsymbol{x}) = \begin{bmatrix} \exp(-\|\boldsymbol{x} - \boldsymbol{x}^1\|^2) \\ \vdots \\ \exp(-\|\boldsymbol{x} - \boldsymbol{x}^N\|^2) \end{bmatrix}$$
- Set only the max component to 1: $\phi'_k(x) = \begin{cases} 1\,, \phi_k(x) = \max \phi(x) \\ 0\,, \text{else} \end{cases}$
- With weight vector
$$\boldsymbol{\theta} = [y^{(1)}, \ldots, y^{(N)}]^T$$
- This is a nearest neighbor classifier $h_\theta(x) = g(\boldsymbol{\theta}^T \phi'(\boldsymbol{x}))$

Different point of view on nearest neighbor

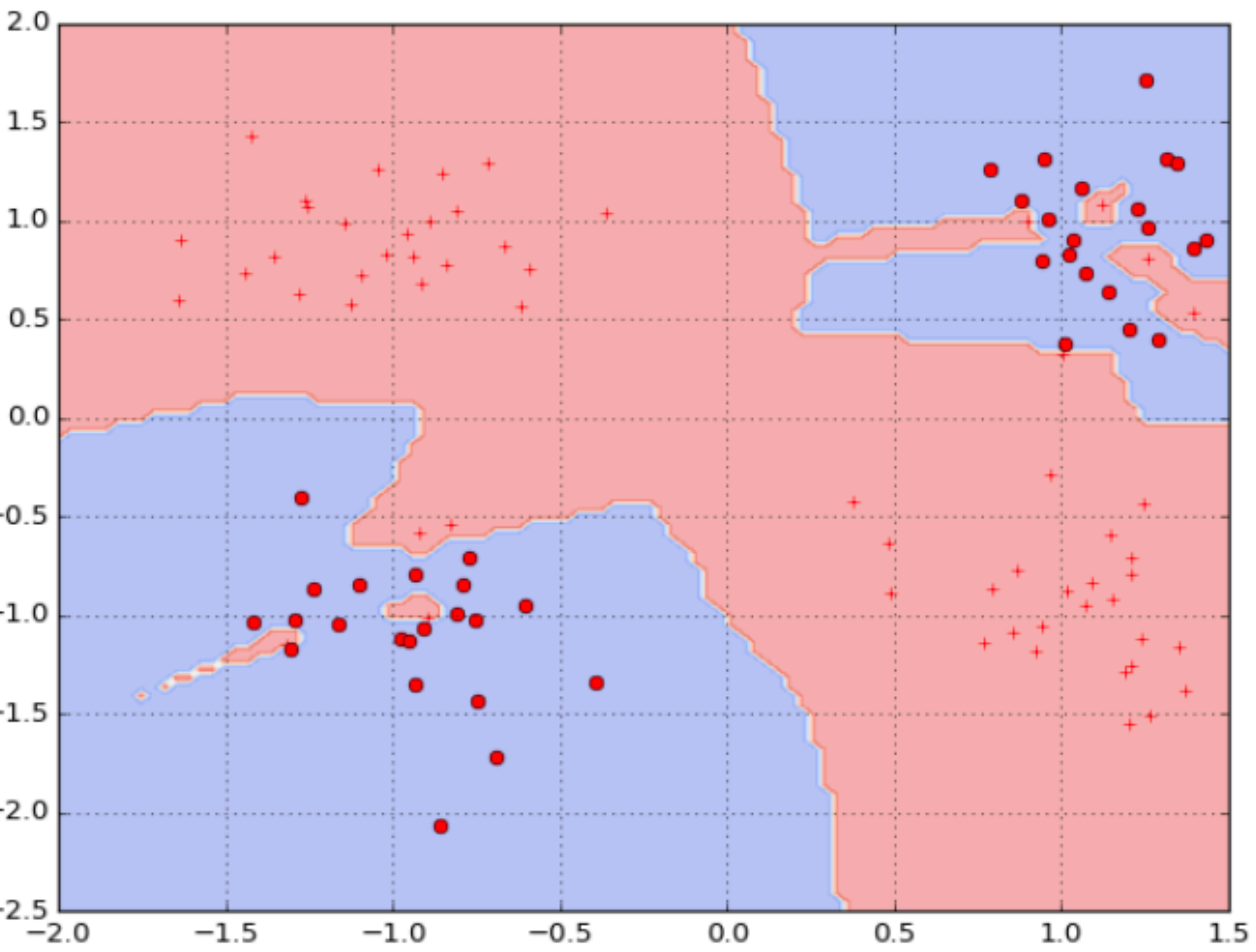
- Instead of feature map, search for nearest neighbor and return label of nearest neighbor
- Different distance than Euclidean $||\cdot||_2$ possible
- Rarely use ``nearest neighbor”. Rather use k-nearest neighbor (KNN)
  - Search k nearest neighbors
  - Vote which label new point gets

- How can we interpret number of neighbors k?

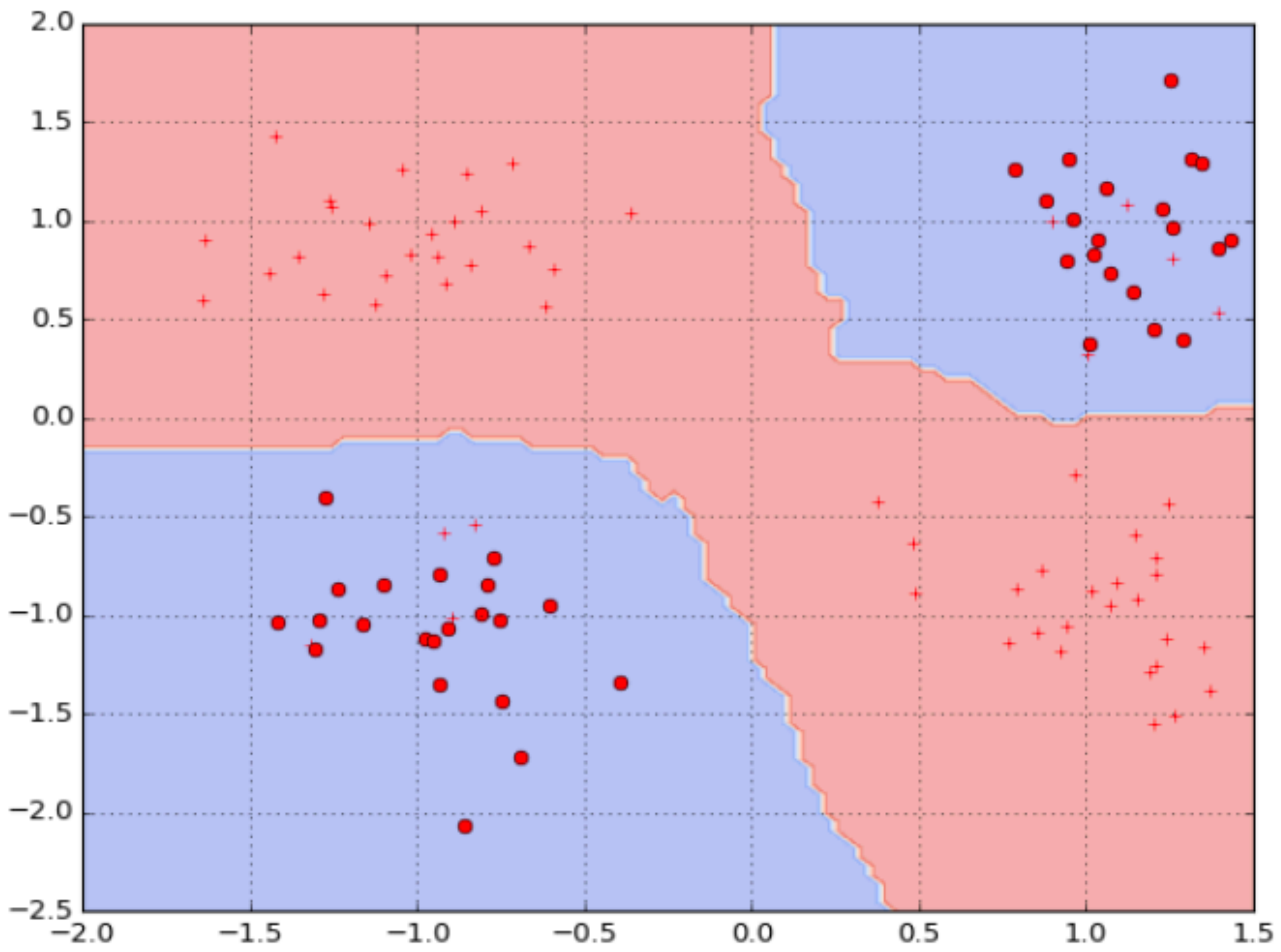
Different point of view on nearest neighbor

- Instead of feature map, search for nearest neighbor and return label of nearest neighbor
- Different distance than Euclidean $||\cdot||_2$ possible
- Rarely use ``nearest neighbor". Rather use k-nearest neighbor (KNN)
  - Search k nearest neighbors
  - Vote which label new point gets
- The number of neighbors serves as regularizer
  - If $k = 1$ then costs low because only single neighbor required
  - However, leads to overfitting because a single point determines label in neighborhood
  - Other extreme is $k = N$, then over-regularization

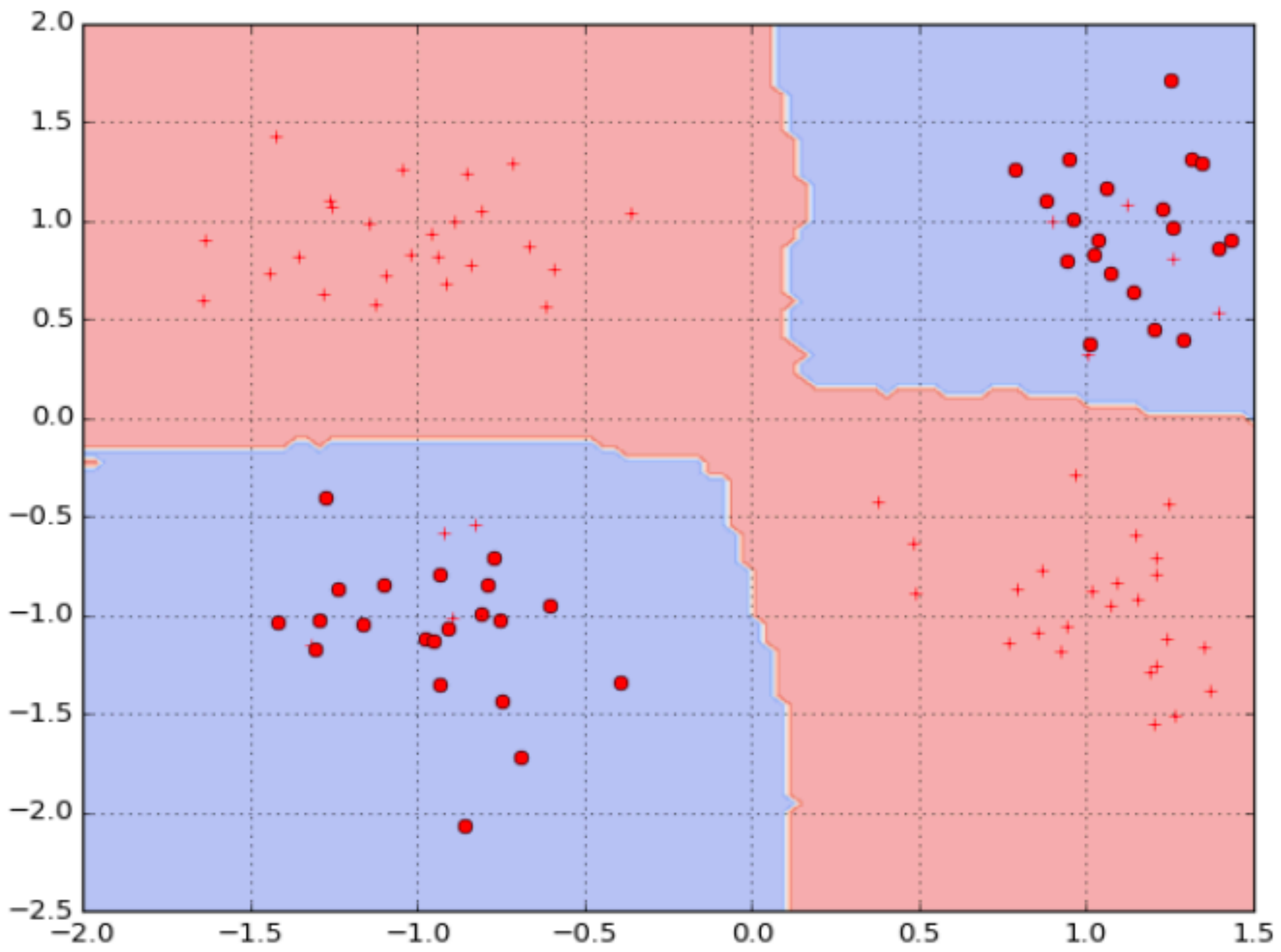
Parameter k controls regularization



(a) $k = 1$



(b) $k = 5$



(c) $k = 20$

As k increases, the decision boundary becomes smoother

Figure: Cho

Selecting basis vectors - random

- Weakness of KNN
  - Requires potentially large storage (because need to store all N training points)
  - Need to sweep through entire training set for each predict
- Idea: Select only a few representative basis vectors
  - Uniform-randomly select an index from $i \in \{1, \ldots, N\}$
  - Set $B = B \cup \{i\}$
  - Set $D = D \cup \{\boldsymbol{x}^{(i)}\}$
  - Repeat if $|B| < k$
- Works surprisingly well
  - Basis vectors are close to training data set
  - Basis vectors "evenly" distributed in training data set (uniform sampling)
  - Weight vector $\boldsymbol{\theta}$ without optimization (cheap)

Selecting basis vectors - clustering

- Earlier in the XOR problem, we first performed clustering and then picked for each cluster a representative basis vector
  - Difficult to find clustering non-visually in >3D
  - Will discuss clustering in detail later
- Once clusters have been selected, need to set weight vector θ
  - The k basis vectors $r^1, \ldots, r^k$ define ϕ

$$\phi(\boldsymbol{x}) = \begin{bmatrix} \exp(-\|\boldsymbol{x} - \boldsymbol{r}^1\|^2) \\ \vdots \\ \exp(-\|\boldsymbol{x} - \boldsymbol{r}^k\|^2) \end{bmatrix}$$

  - **No labels** for basis vectors $r^1, \ldots, r^k$ given
    - Gives new training data set $D_\phi = \{(\phi(x^{(1)}), y^{(1)}), \ldots, (\phi(x^{(N)}), y^{(N)})\}$
    - Apply ***linear*** classifier to D_ϕ (logistic regression, SVM, etc)
  - Sometimes known under name “radial basis function network”

Learning the feature map (towards deep learning)

- So far, we said given are basis vectors that define feature map
- Then, we found the weights via linear classification
- In principle, we can parametrize the basis vectors as well
  and then optimize for the right basis vectors via gradient descent
- Further, we could use another ϕ than the one we used based on radial basis functions
  - In principle, any differentiable ϕ is feasible
  - We could think of one ϕ that does the feature map in one shot
  - Or we could compose many simpler ones

$$\phi = \phi_\ell(\cdots\phi_1(x)) = (\phi_\ell \circ \ldots \circ \phi_1)(x)$$

  - This is a topic of deep learning $\Rightarrow$ see other courses