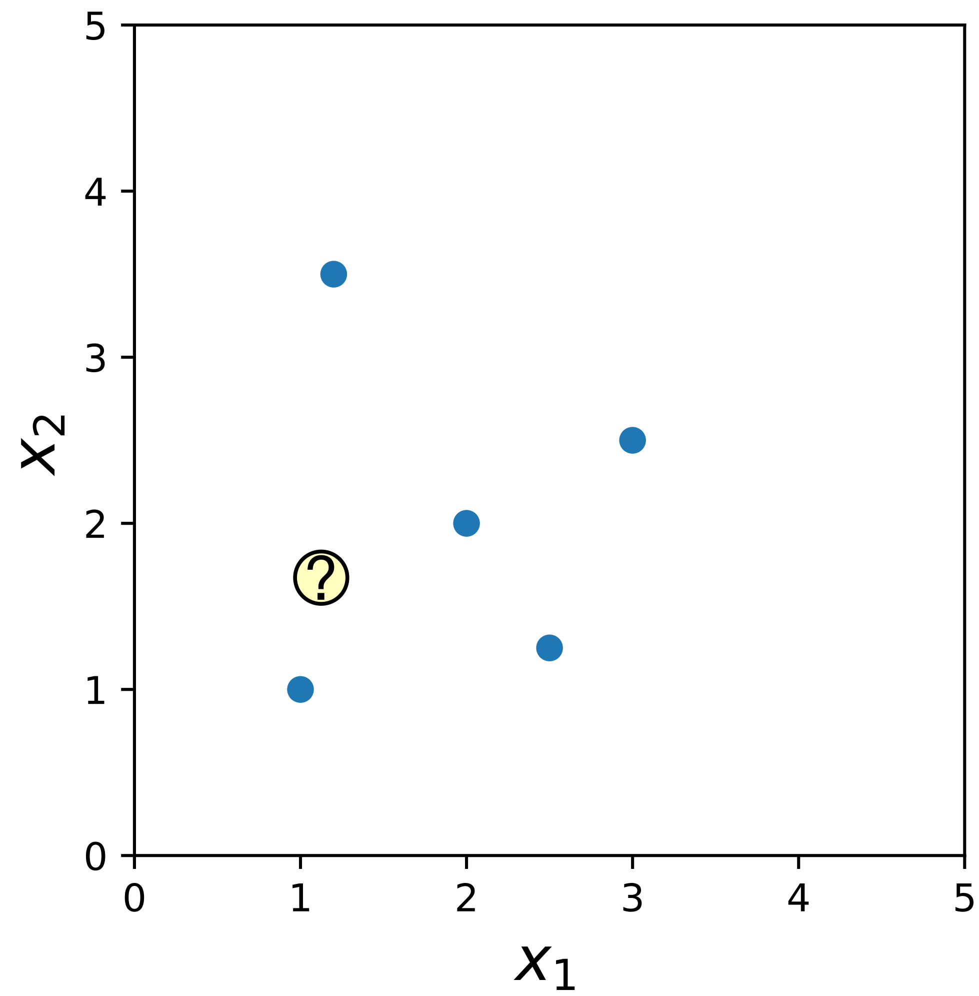
Lecture 02

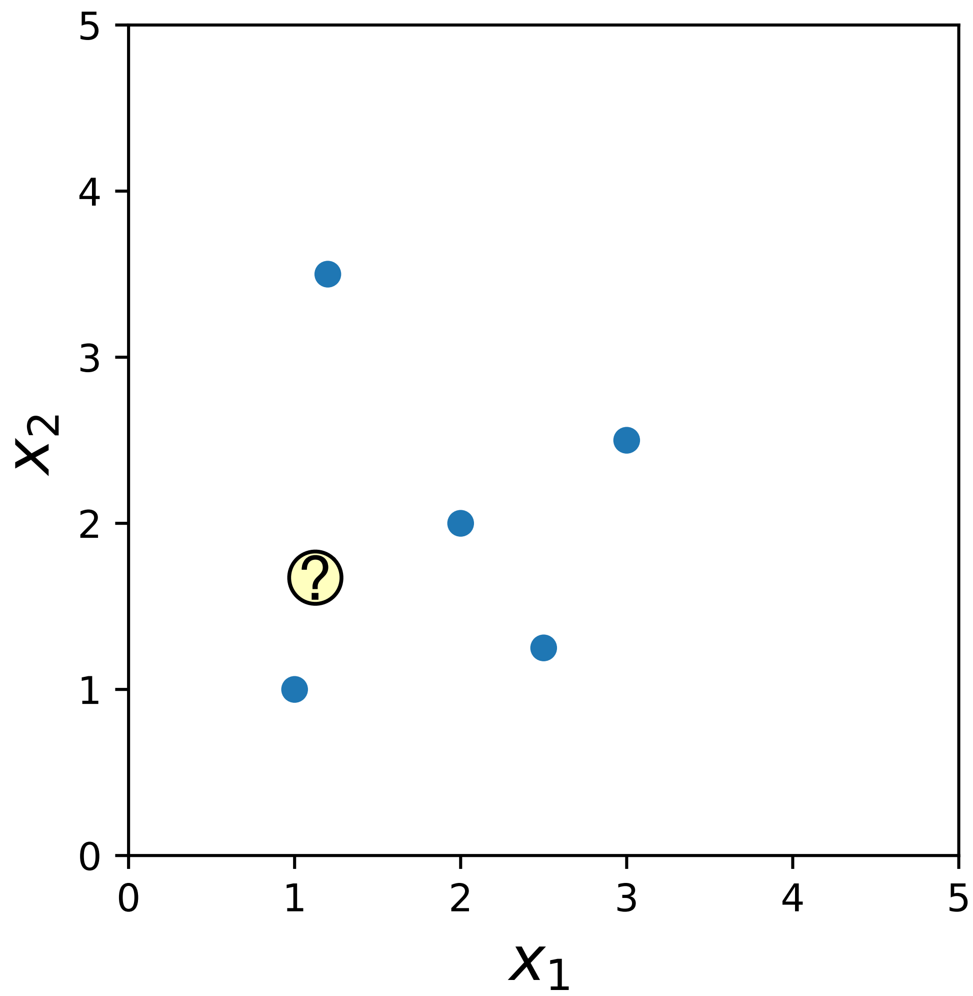
Nearest Neighbor Methods

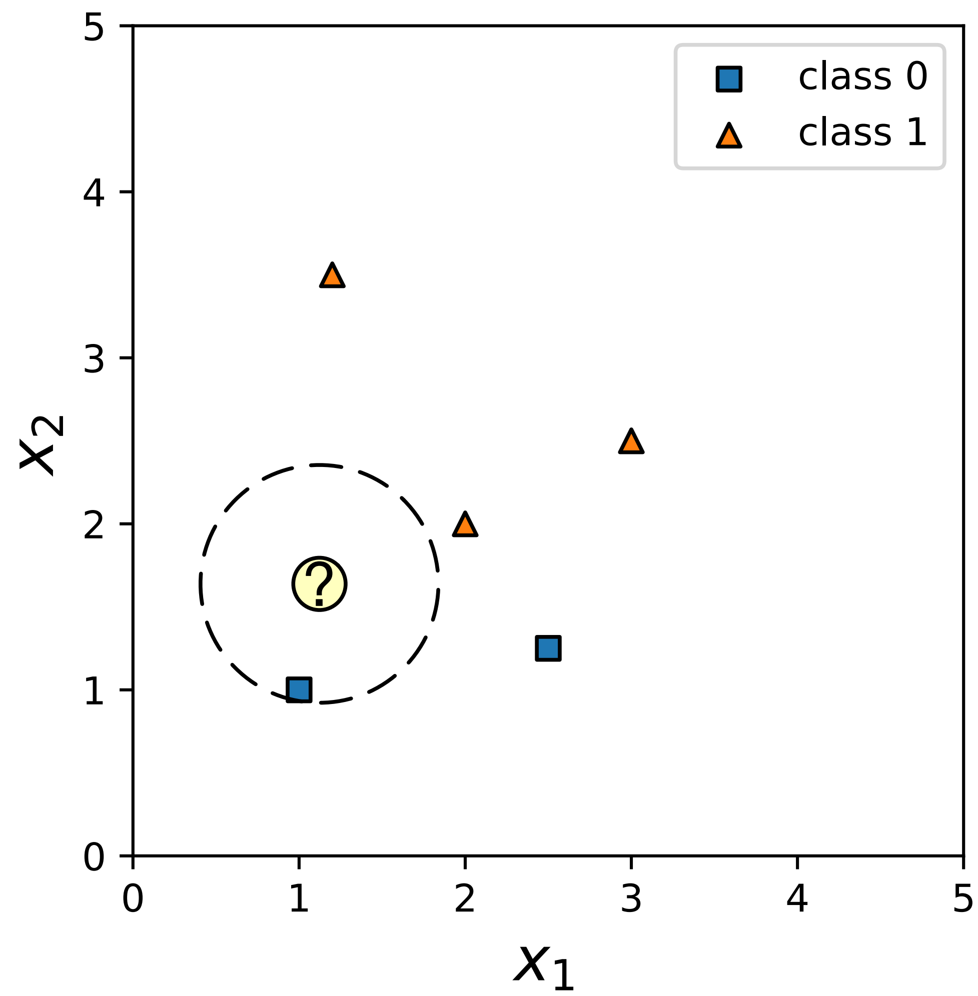
STAT 479: Machine Learning, Fall 2018
Sebastian Raschka
http://stat.wisc.edu/~sraschka/teaching/stat479-fs2018/

1-Nearest Neighbor



1-Nearest Neighbor





Training Step

$$\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathscr{D} \quad (\,|\,\mathscr{D}\,| = n)$$

1-Nearest Neighbor Prediction Step

closest_point := None

closest_distance := ∞

- for $i = 1, \ldots, n$:
  - current_distance := $d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if current_distance < closest_distance:
    - closest_distance := current_distance
    - closest_point := $\mathbf{x}^{[i]}$
- return f(closest_point)

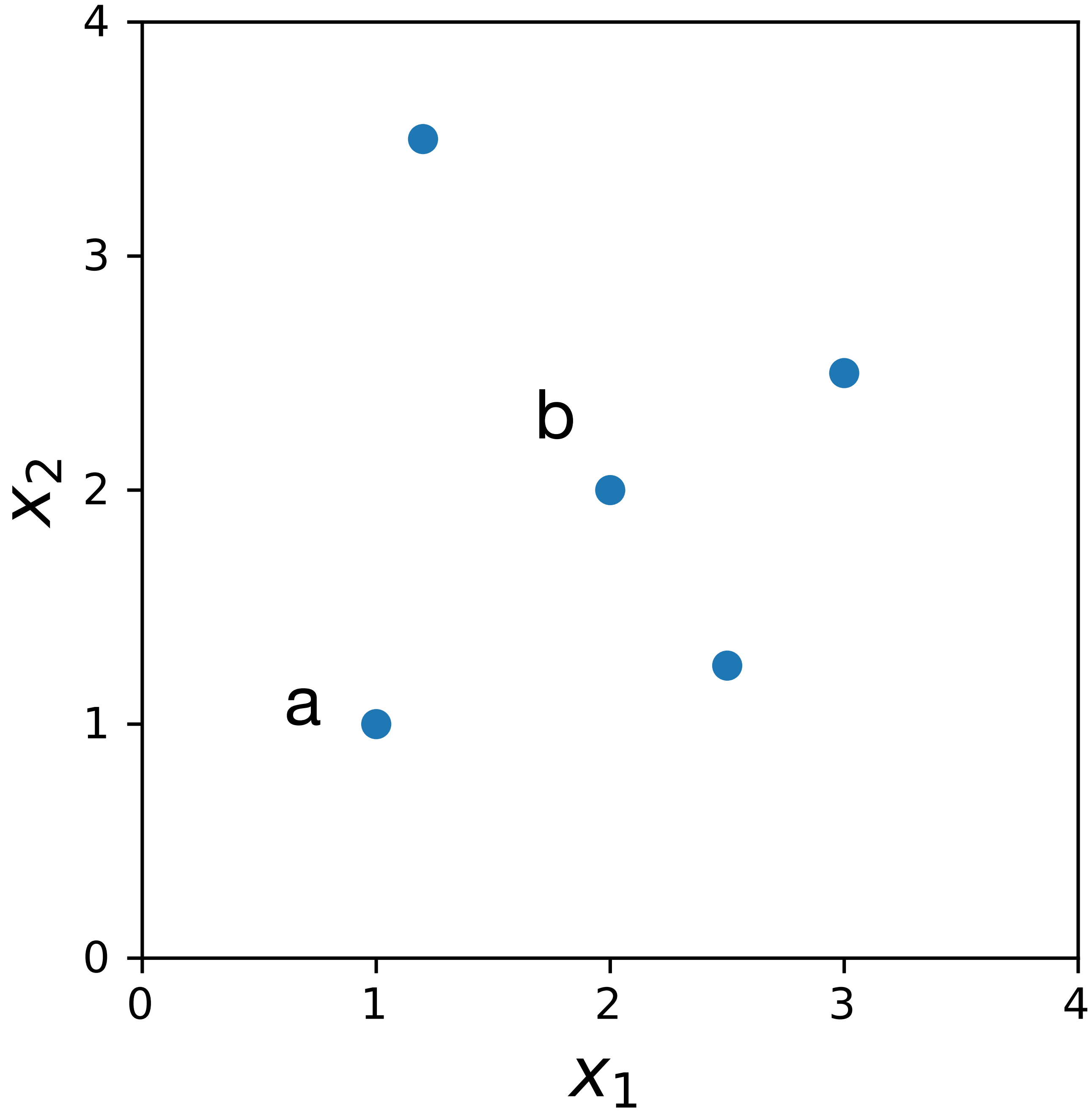
closest_point is the label of $\langle \mathbf{x}^{[i]}, f\big(\mathbf{x}^{[i]}\big) \rangle$

Commonly used: Euclidean Distance (*L*²)

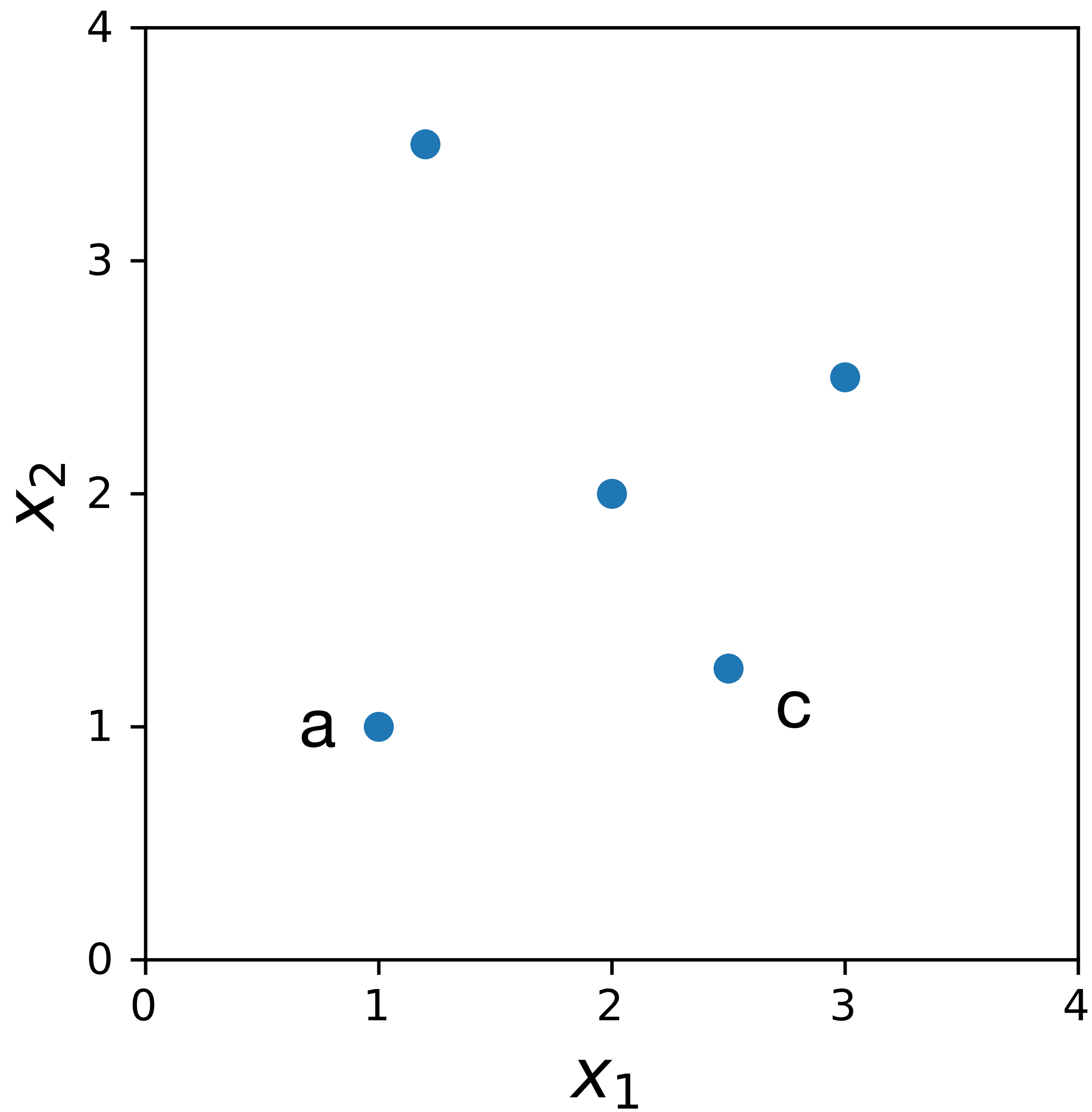
$$d(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \sqrt{\sum_{j=1}^{m} \left(x_j^{[a]} - x_j^{[b]} \right)^2}$$

Nearest Neighbor Decision Boundary

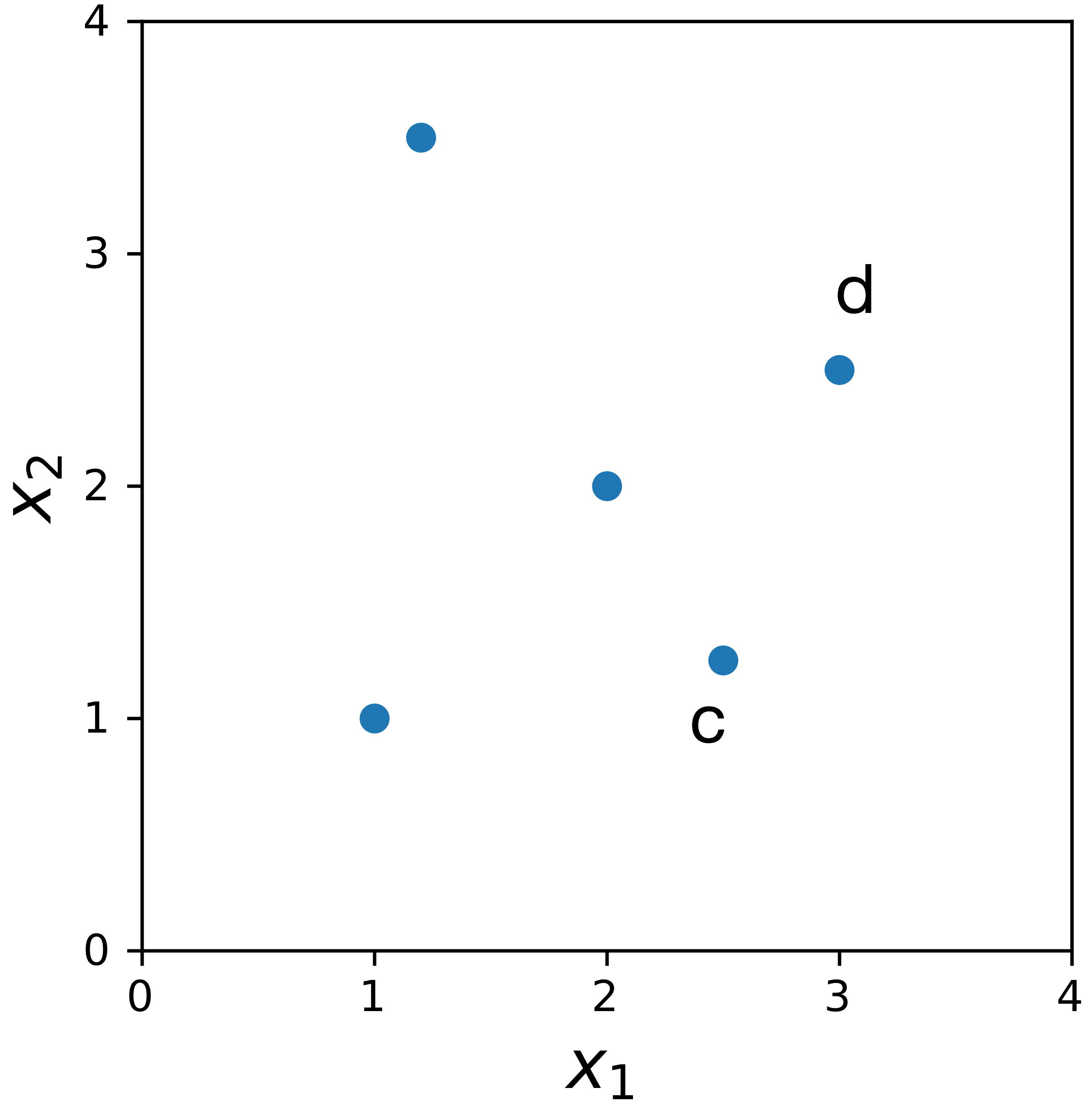
Decision Boundary Between (a) and (b)



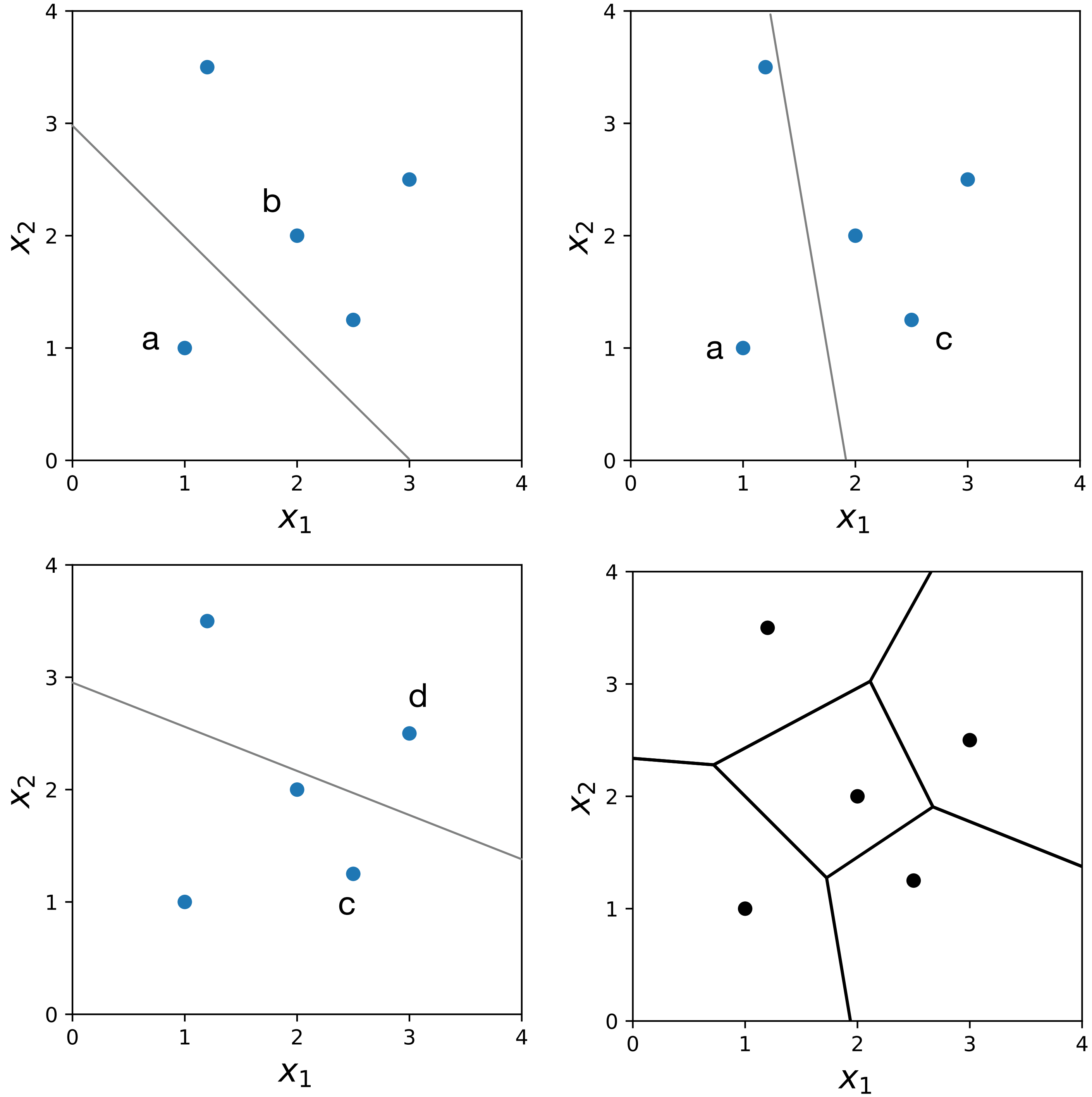
Decision Boundary Between (a) and (c)



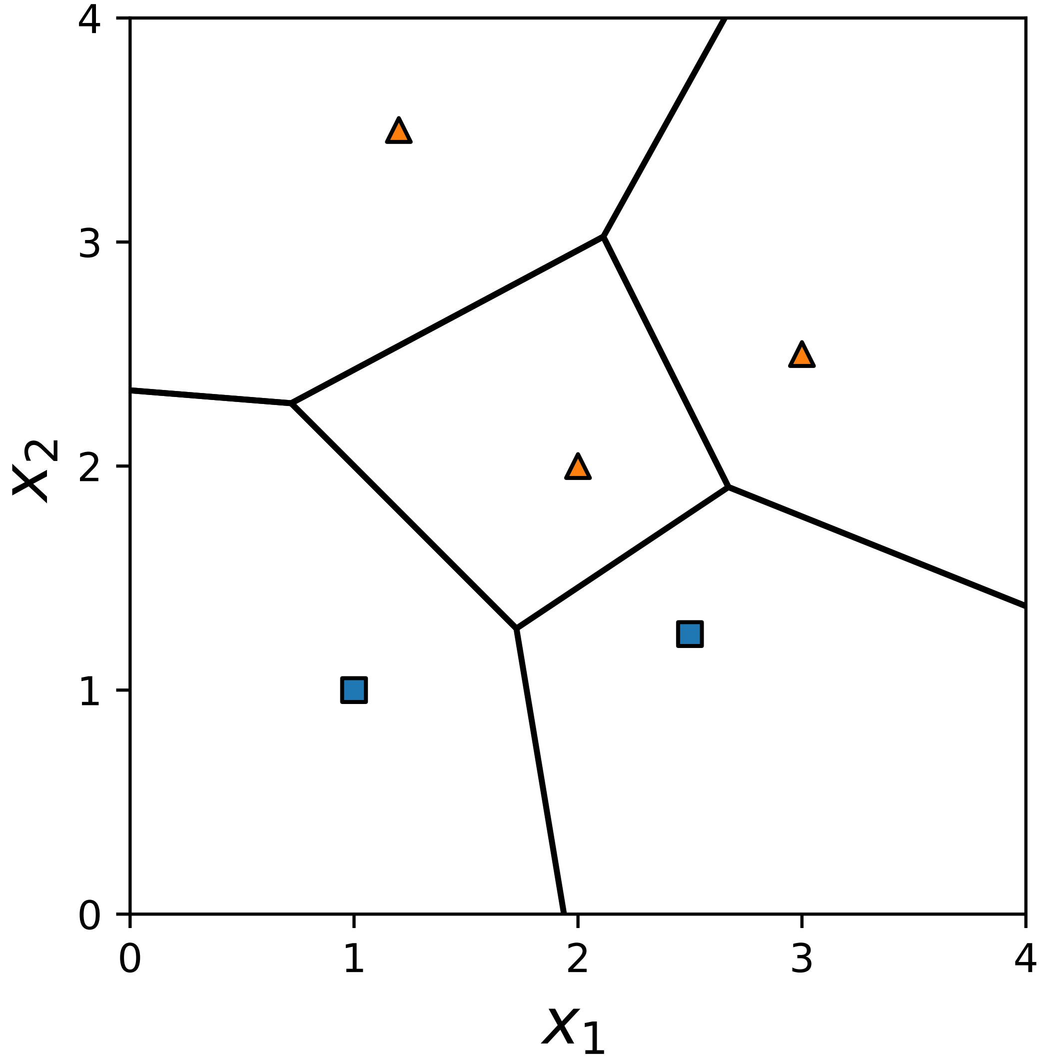
Decision Boundary Between (a) and (c)

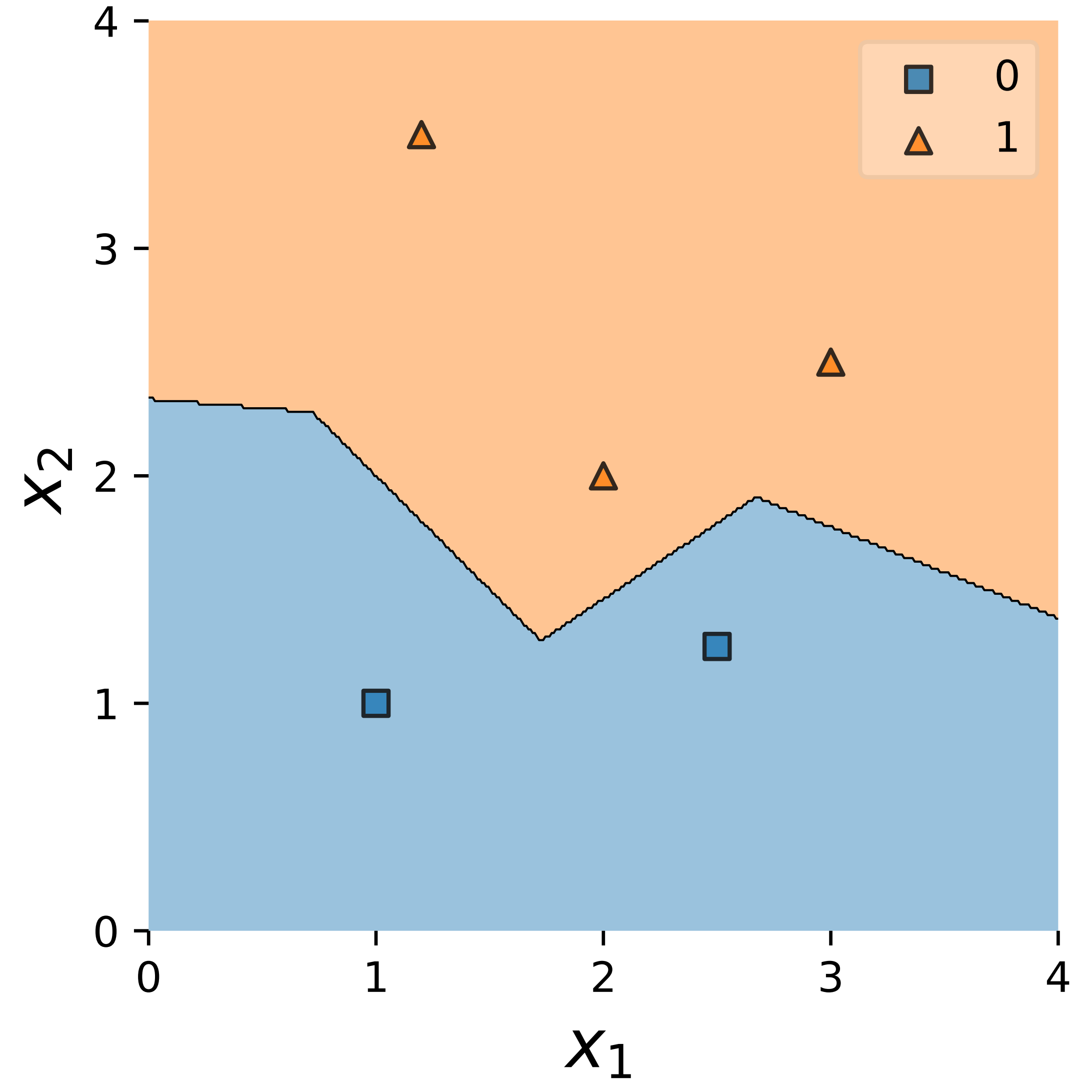


Decision Boundary 1NN

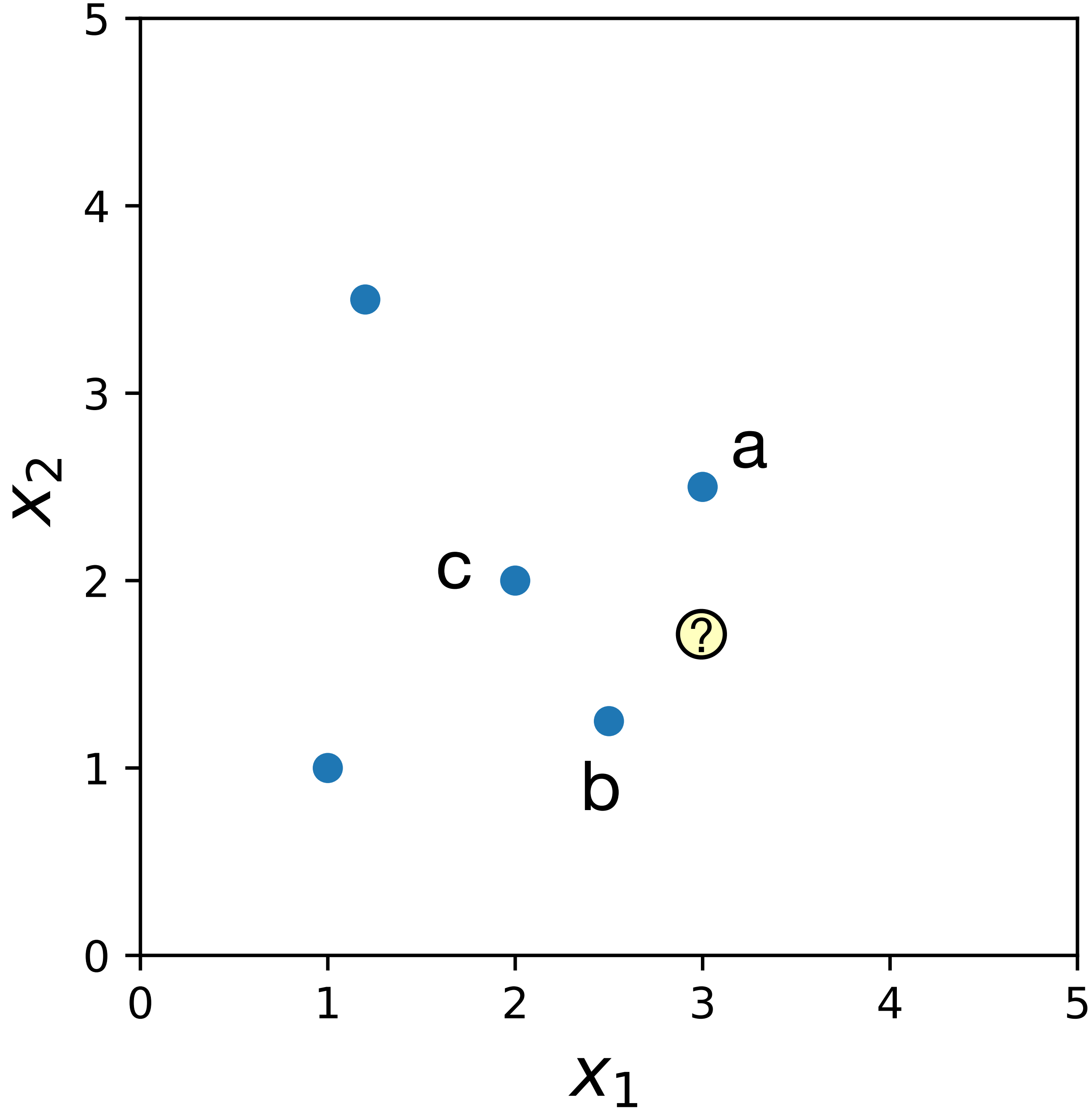


Decision Boundary 1-NN

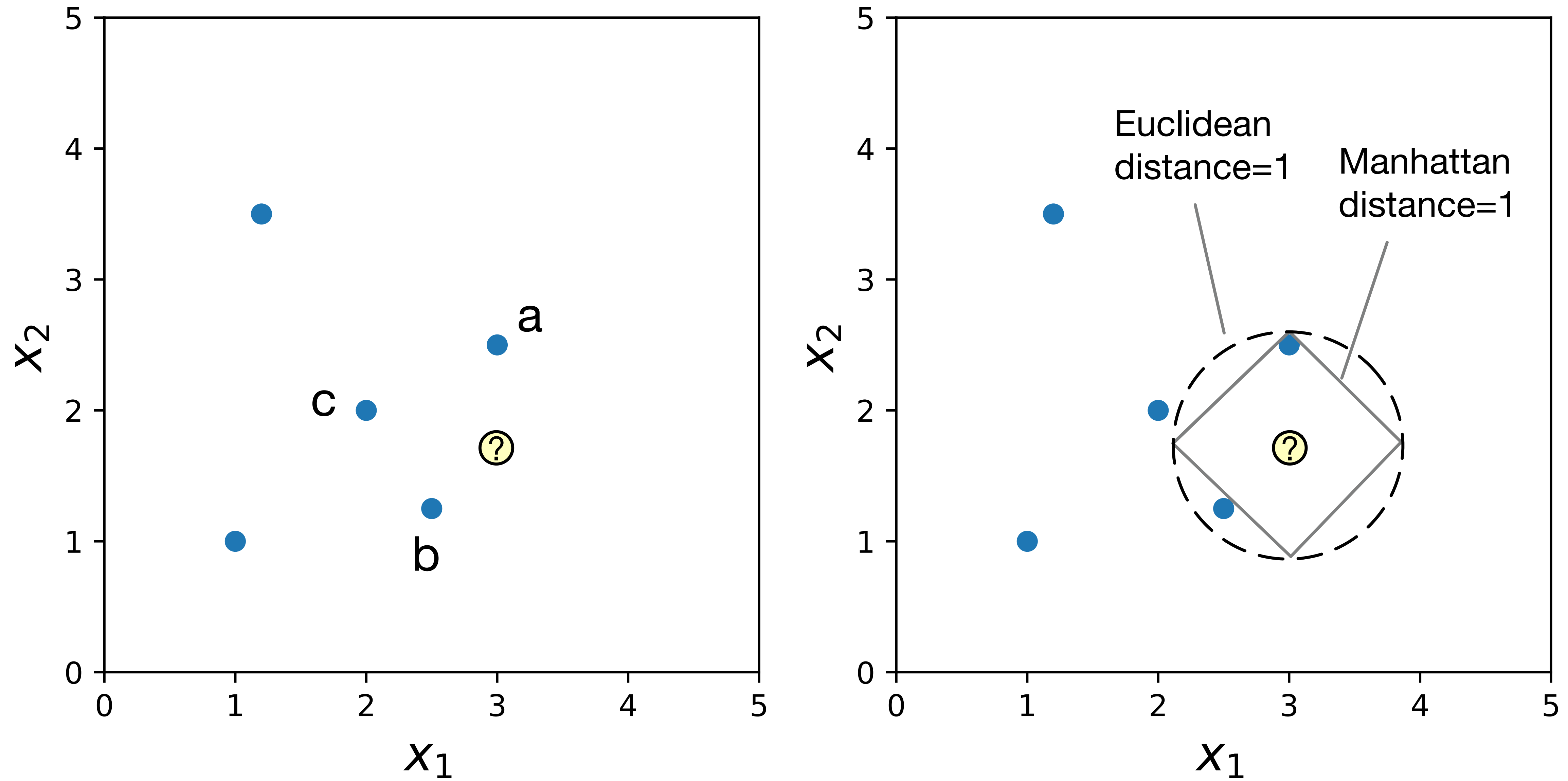




Which Point is Closest?



Depends on the Distance Measure!



Continuous Distance Measures

Euclidean

Manhattan

Minkowski: $d(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \left[\sum_{j=1}^{m} \left(\left| x^{[a]} - x^{[b]} \right| \right)^{p} \right]^{\frac{1}{p}}$

Mahalanobis

...

Discrete Distance Measures

Hamming: $d(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \sum_{j=1}^{m} \left| x^{[a]} - x^{[b]} \right|$

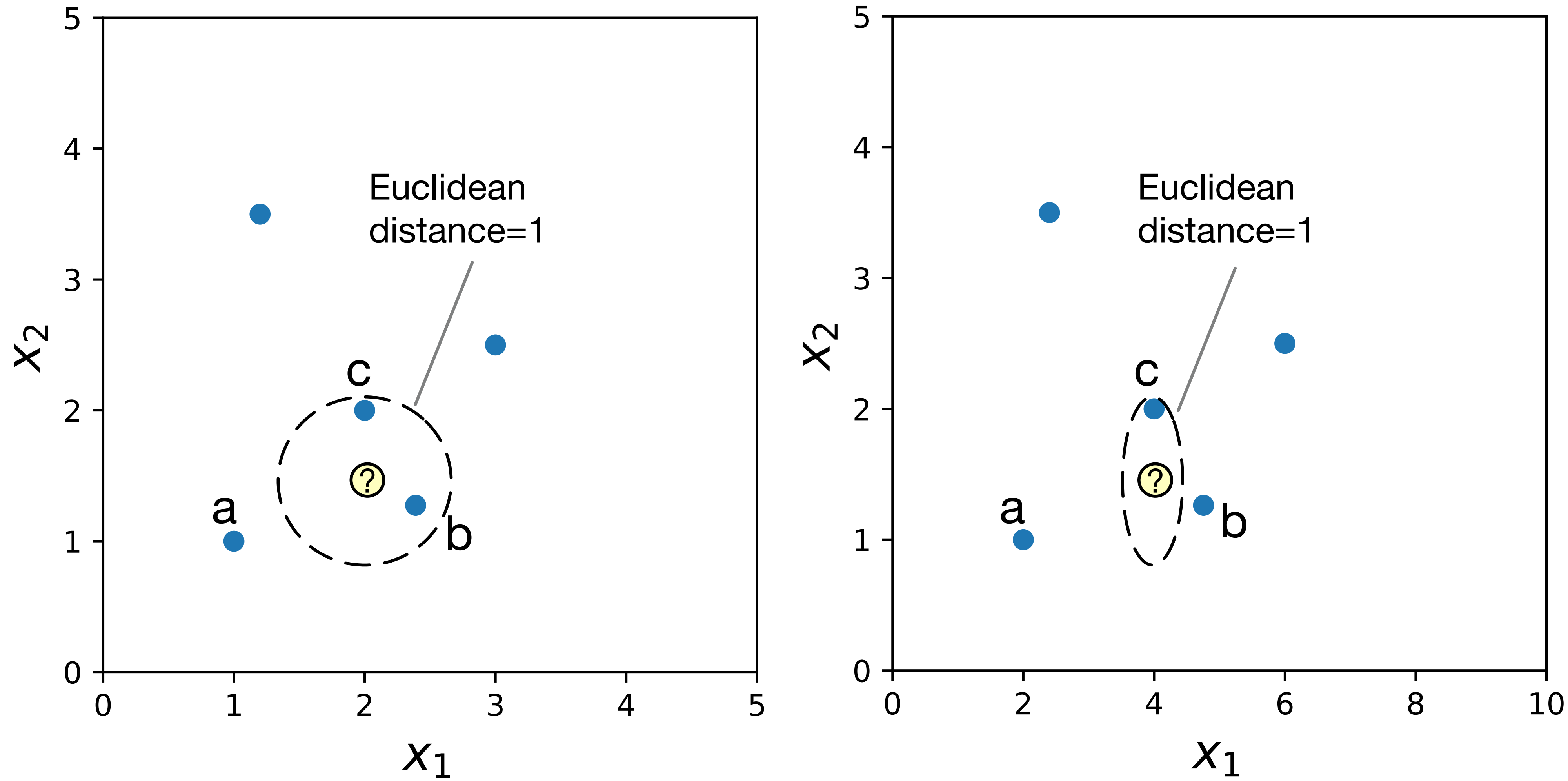
Jaccard/Tanimoto

Cosine similarity

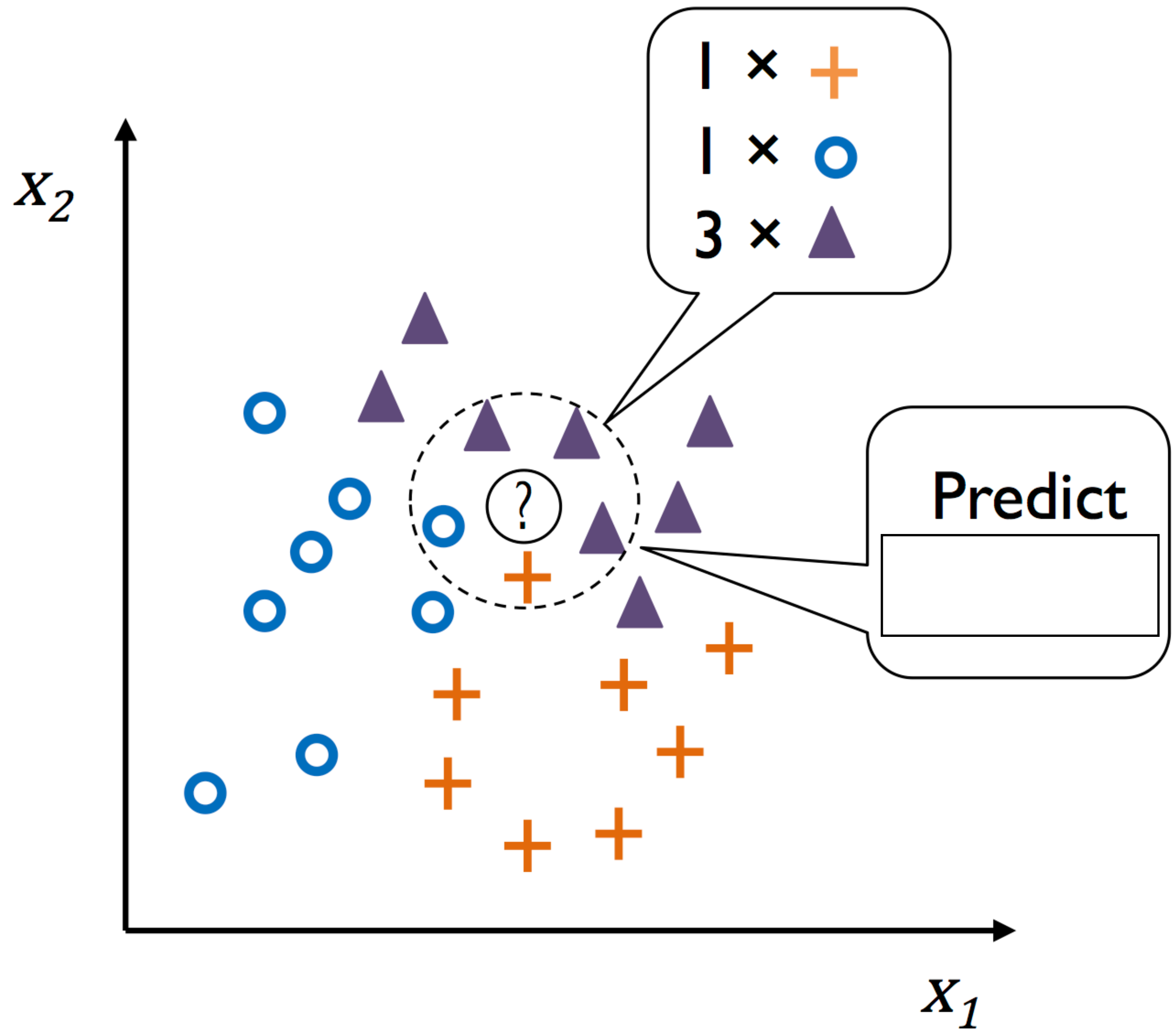
Dice

...

Feature Scaling



k-Nearest Neighbors



A

y: 

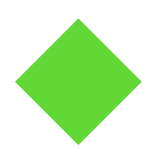
Majority vote: 

Purality vote: 

B

y: 

Majority vote: None

Purality vote: 

kNN for Classification

$$\mathscr{D}_k = \{\langle \mathbf{x}^{[1]}, f\big(\mathbf{x}^{[1]}\big)\rangle, \ldots, \langle \mathbf{x}^{[k]}, f\big(\mathbf{x}^{[k]}\big)\rangle\} \qquad \mathscr{D}_k \subseteq \mathscr{D}$$

*k*NN for Classification

$$\mathscr{D}_k = \{\langle \mathbf{x}^{[1]}, f\big(\mathbf{x}^{[1]}\big)\rangle, \ldots, \langle \mathbf{x}^{[k]}, f\big(\mathbf{x}^{[k]}\big)\rangle\} \qquad \mathscr{D}_k \subseteq \mathscr{D}$$

$$h(\mathbf{x}^{[q]}) = arg \max_{y \in \{1,\ldots,t\}} \sum_{i=1}^{k} \delta(y, f(\mathbf{x}^{[i]}))$$

$$\delta(a, b) = \begin{cases} 1, & \textbf{if } a = b, \\ 0, & \textbf{if } a \neq b\,. \end{cases}$$

*k*NN for Classification

$$\mathcal{D}_k = \{\langle \mathbf{x}^{[1]}, f\big(\mathbf{x}^{[1]}\big)\rangle, \ldots, \langle \mathbf{x}^{[k]}, f\big(\mathbf{x}^{[k]}\big)\rangle\} \qquad \mathcal{D}_k \subseteq \mathcal{D}$$

$$h(\mathbf{x}^{[q]}) = arg \max_{y \in \{1,\ldots,t\}} \sum_{i=1}^{k} \delta(y, f(\mathbf{x}^{[i]}))$$

$$\delta(a, b) = \begin{cases} 1, & \textbf{if } a = b, \\ 0, & \textbf{if } a \neq b\,. \end{cases}$$

$$h(\mathbf{x}^{[t]}) = \textbf{mode}\Big(\big\{f\big(\mathbf{x}^{[1]}\big), \ldots, f\big(\mathbf{x}^{[k]}\big)\big\}\Big)$$

*k*NN for Regression

$$\mathcal{D}_k = \{\langle \mathbf{x}^{[1]}, f(\mathbf{x}^{[1]}) \rangle, \ldots, \langle \mathbf{x}^{[k]}, f(\mathbf{x}^{[k]}) \rangle\} \qquad \mathcal{D}_k \subseteq \mathcal{D}$$

$$h(\mathbf{x}^{[t]}) = \frac{1}{k} \sum_{i=1}^{k} f(\mathbf{x}^{[i]})$$

Categories (Last Lecture)

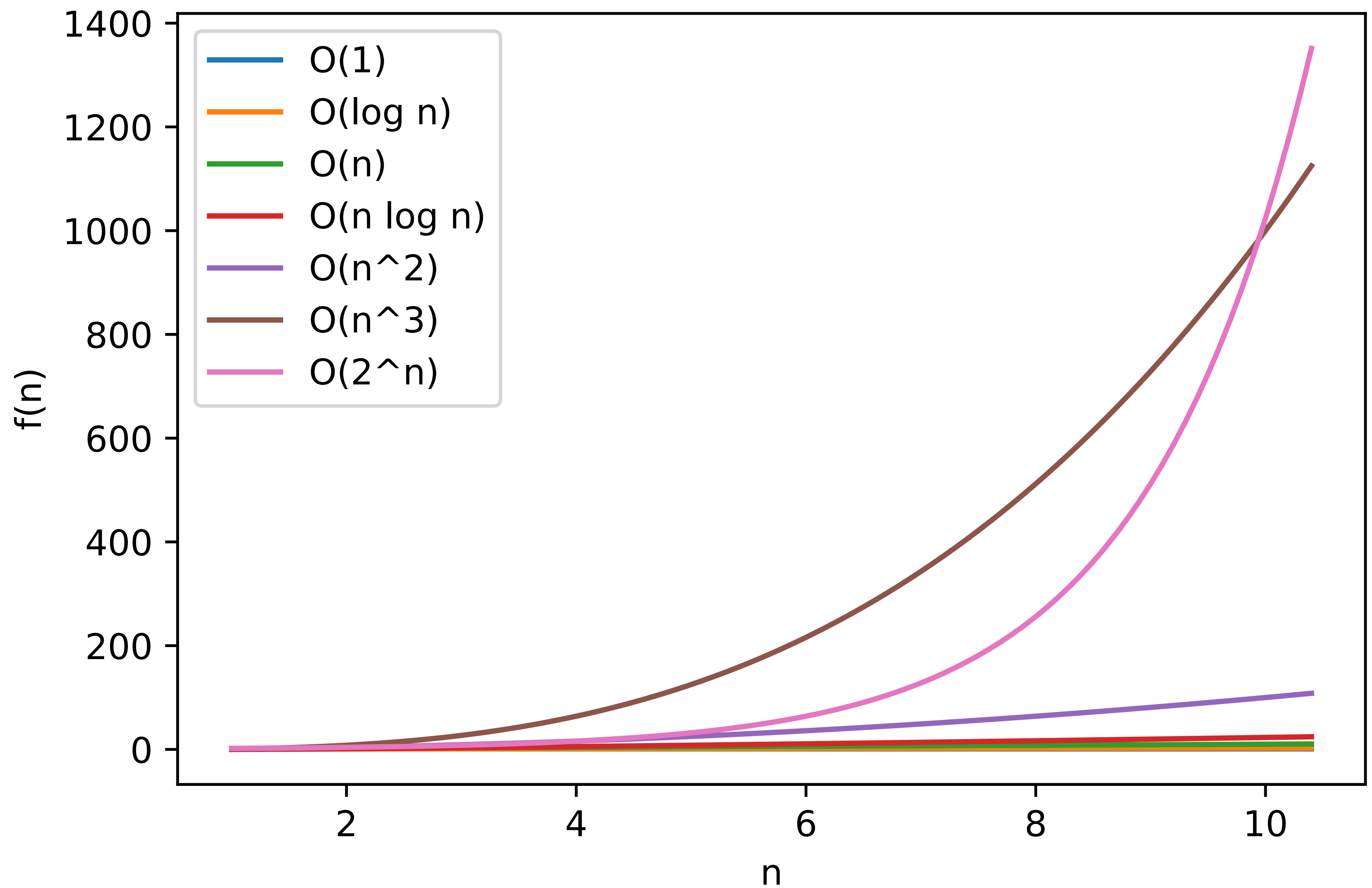
- eager vs lazy;
- batch vs online;
- parametric vs nonparametric;
- discriminative vs generative.

Big-O

f(n)	**Name**
1	Constant
$\log n$	Logarithmic
n	Linear
$n \log n$	Log Linear
n^2	Quadratic
n^3	Cubic
n^c	Higher-level polynomial
2^n	Exponential

Big-O

f(n)	Name
1	Constant
$\log n$	Logarithmic
n	Linear
$n \log n$	Log Linear
n^2	Quadratic
n^3	Cubic
n^c	Higher-level polynomial
2^n	Exponential



Big-O Example 1

$$f(x) = 14x^2 - 10x + 25$$

Big-O Example 2

$$f(x) = (2x + 8)\log_2(x + 9)$$

Big-O Example 3

```
A = [[1, 2, 3],
     [2, 3, 4]]

B = [[5, 8],
     [6, 9],
     [7, 10]]


def matrixmultiply (A, B):

    C = [[0 for row in range(len(A))]
          for col in range(len(B[0]))]

    for row_a in range(len(A)):
        for col_b in range(len(B[0])):
            for col_a in range(len(A[0])):
                C[row_a][col_b] += \
                    A[row_a][col_a] * B[col_a][col_b]
    return C

matrixmultiply(A, B)
```

```
Out[16]:

[[38, 56], [56, 83]]
```

Big O of *k*NN

Improving Computational Performance

Naive Nearest Neighbor Search

Variant A

$O(\ \_\_\_\_\_\ )$

$\mathcal{D}_k := \{\}$

while $|\mathcal{D}_k| <$ k:

- `closest_distance` $:= \infty$
- for $i = 1, ..., n, \quad \forall i \notin \mathcal{D}_k$:
  - `current_distance` $:= d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if `current_distance` $<$ `closest_distance`:
    - `closest_distance` $:=$ `current_distance`
    - `closest_point` $:= \mathbf{x}^{[i]}$
- add `closest_point` to $\mathcal{D}_k$

Naive Nearest Neighbor Search

$O(\_\_\_\_)$

Variant B

$\mathcal{D}_k := \mathcal{D}$

while $|\mathcal{D}_k| > k$:

- `largest_distance` $:= 0$
- for $i = 1, ..., n \quad \forall i \in \mathcal{D}_k$:
  - `current_distance` $:= d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if `current_distance` $>$ `largest_distance`:
    - `largest_distance` := `current_distance`
    - `farthest_point` $:= \mathbf{x}^{[i]}$

Naive Nearest Neighbor Search

Using a priority queue $O($ ____ $)$

Improving Computational Performance

Data Structures

Improving Computational Performance

Dimensionality Reduction

Improving Computational Performance

Editing / "Pruning"

Improving Computational Performance

Prototypes

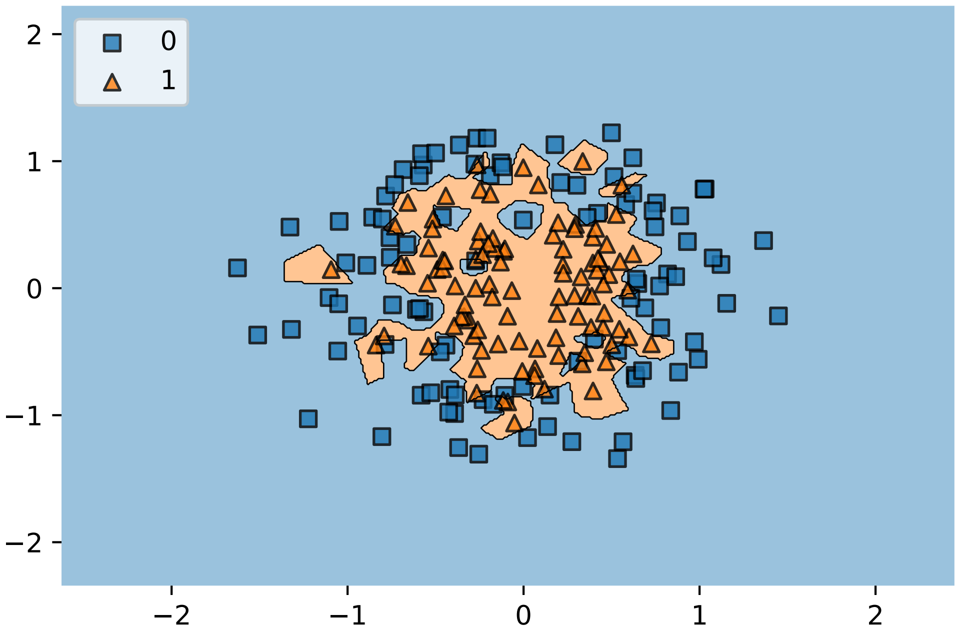
Improving Predictive Performance

Hyperparameters

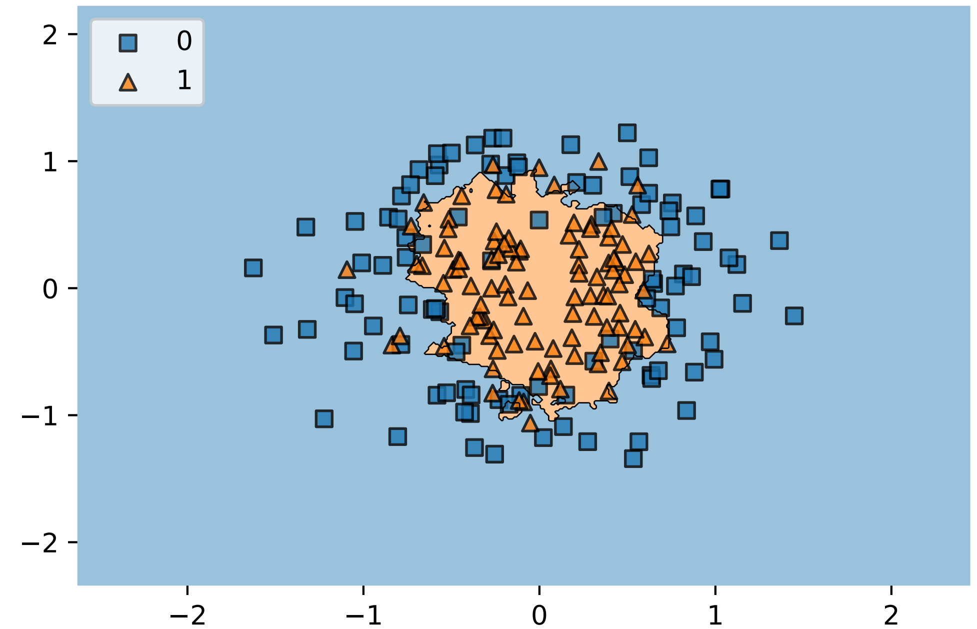
- Value of *k*
- Scaling of the feature axes
- Distance measure
- Weighting of the distance measure

$k \in \{1,3,7\}$

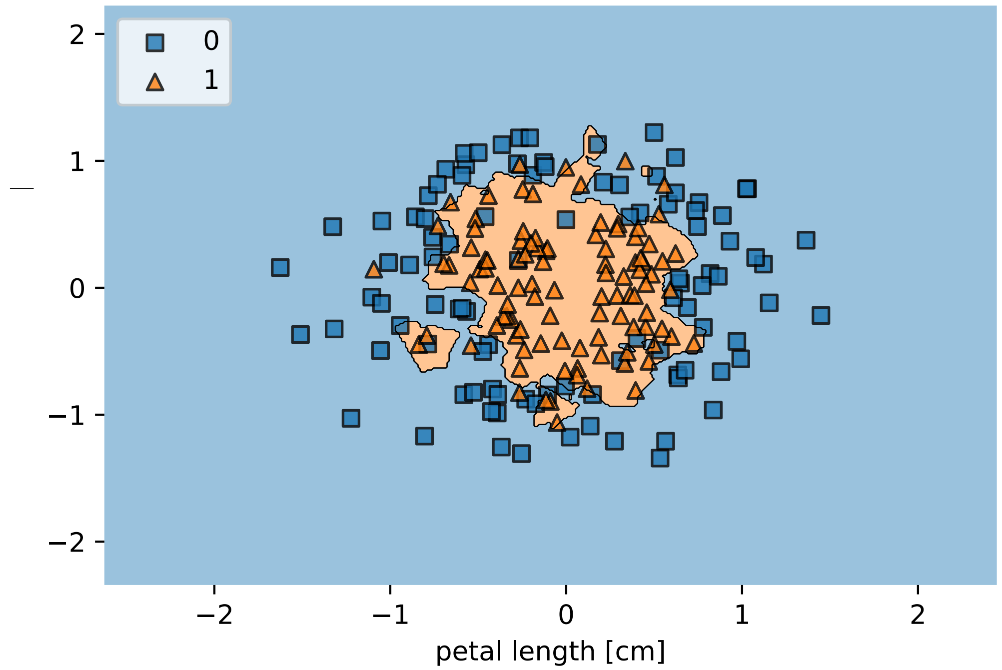
$k = \_$



$k = \_$



$k = \_$



Feature-Weighting via Euclidean Distance

$$d_w(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \sqrt{\sum_{j=1}^{m} w_j \left(x_j^{[a]} - x_j^{[b]}\right)^2}$$

As a dot product:

$$\mathbf{c} = \mathbf{x}^{[a]} - \mathbf{x}^{[a]}, \quad (\mathbf{c}, \mathbf{x}^{[a]}\mathbf{x}^{[b]} \in \mathbb{R}^m)$$

$$d(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \sqrt{\mathbf{c}^t \mathbf{c}}$$

$$d_w(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \mathbf{c}^T \mathbf{W} \mathbf{c}, \qquad \mathbf{W} \in \mathbb{R}^{m \times m} = \textbf{diag}(w_1, w_2, \ldots, w_m)$$

Distance-weighted *k*NN

$$h(\mathbf{x}^{[t]}) = arg \max_{j \in \{1,\ldots,p\}} \sum_{i=1}^{k} w^{[i]} \delta(j, f(\mathbf{x}^{[i]}))$$

$$w^{[i]} = \frac{1}{d(\mathbf{x}^{[i]}, \mathbf{x}^{[t]})^2}$$

Small constant to avoid zero division
or set $h(\mathbf{x}) = f(\mathbf{x})$

*k*NN in Python

DEMO

Reading Assignments

- Lecture notes (will be uploaded after this lecture!)
- Elements of Statistical Learning, Ch 02, Sections 2.0-2.3 (https://web.stanford.edu/~hastie/ElemStatLearn/)

Ungraded Homework Assignment

For those who are new to Python, I highly recommend getting some practice. E.g., by solving some interactive learning exercises on

https://www.codecademy.com/learn/learn-python