

Lecture 02

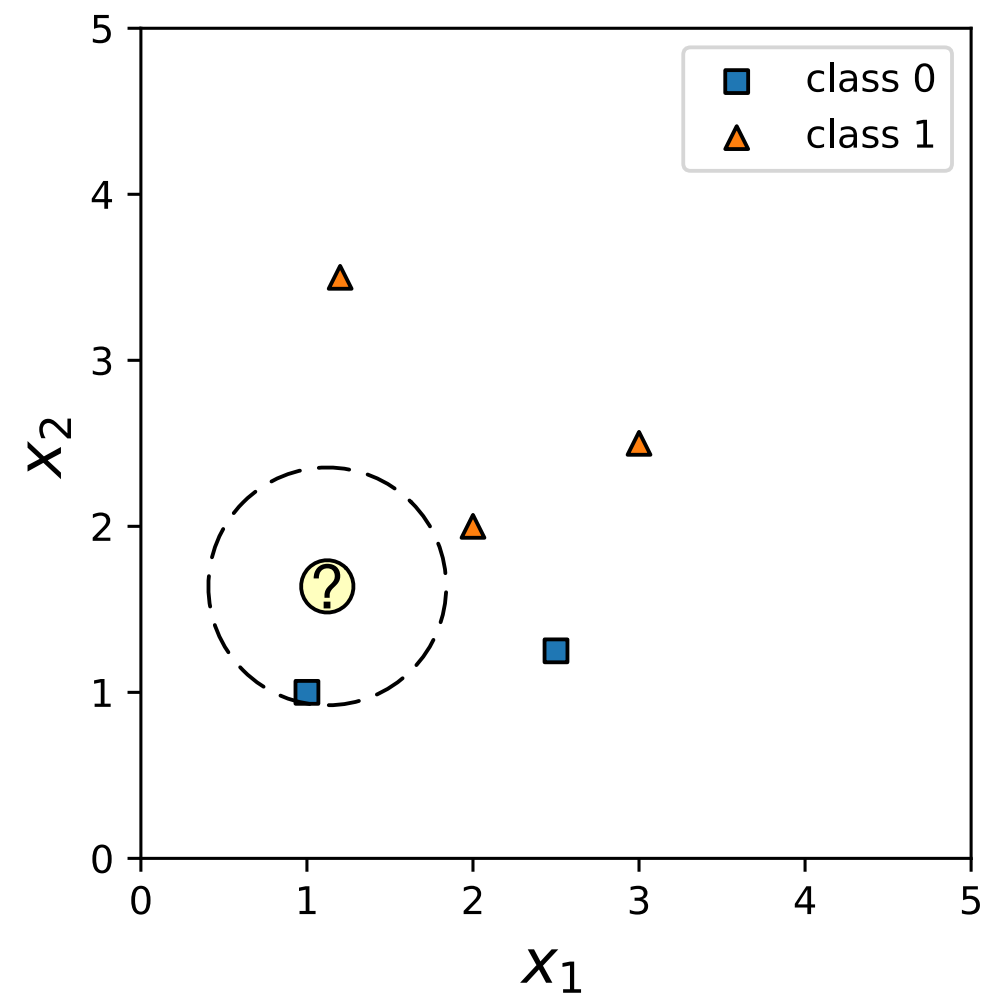
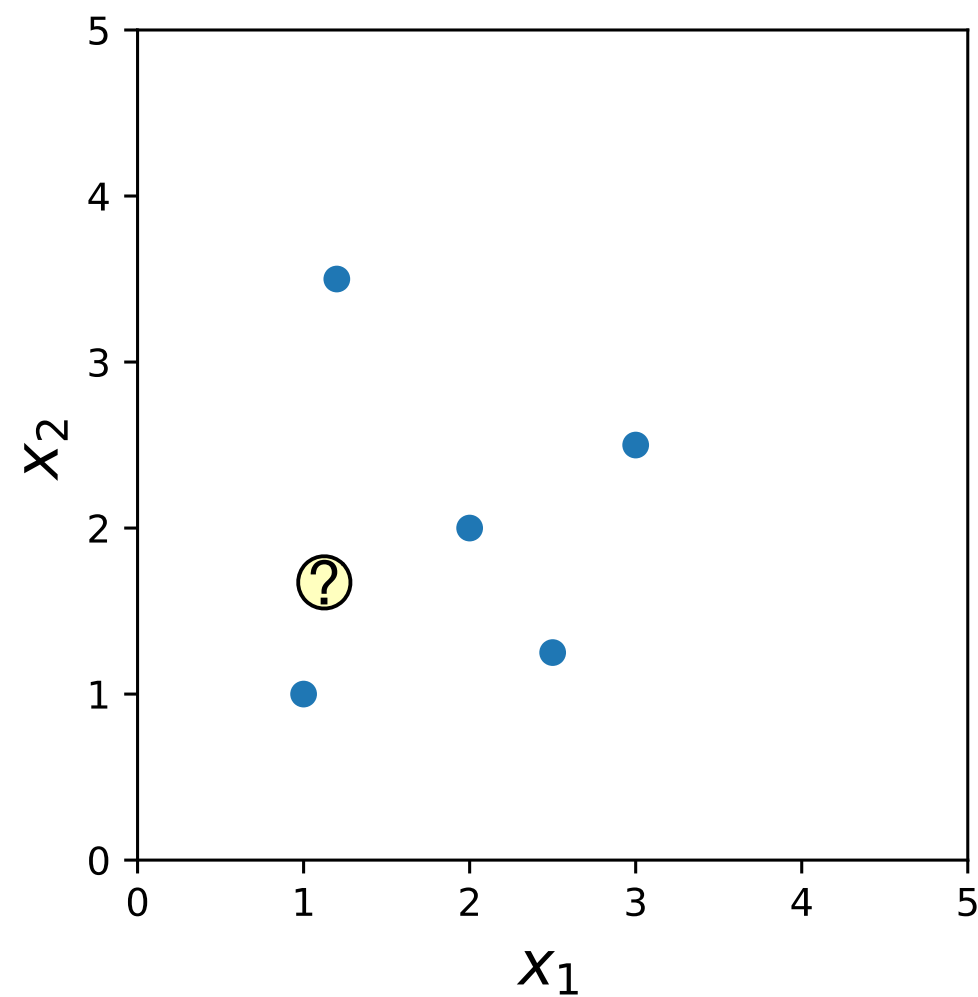
Nearest Neighbor Methods

STAT 479: Machine Learning, Fall 2018

Sebastian Raschka

<http://stat.wisc.edu/~sraschka/teaching/stat479-fs2018/>

1-Nearest Neighbor



Training Step

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

1-Nearest Neighbor Prediction Step

closest_point := None

closest_distance := ∞

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

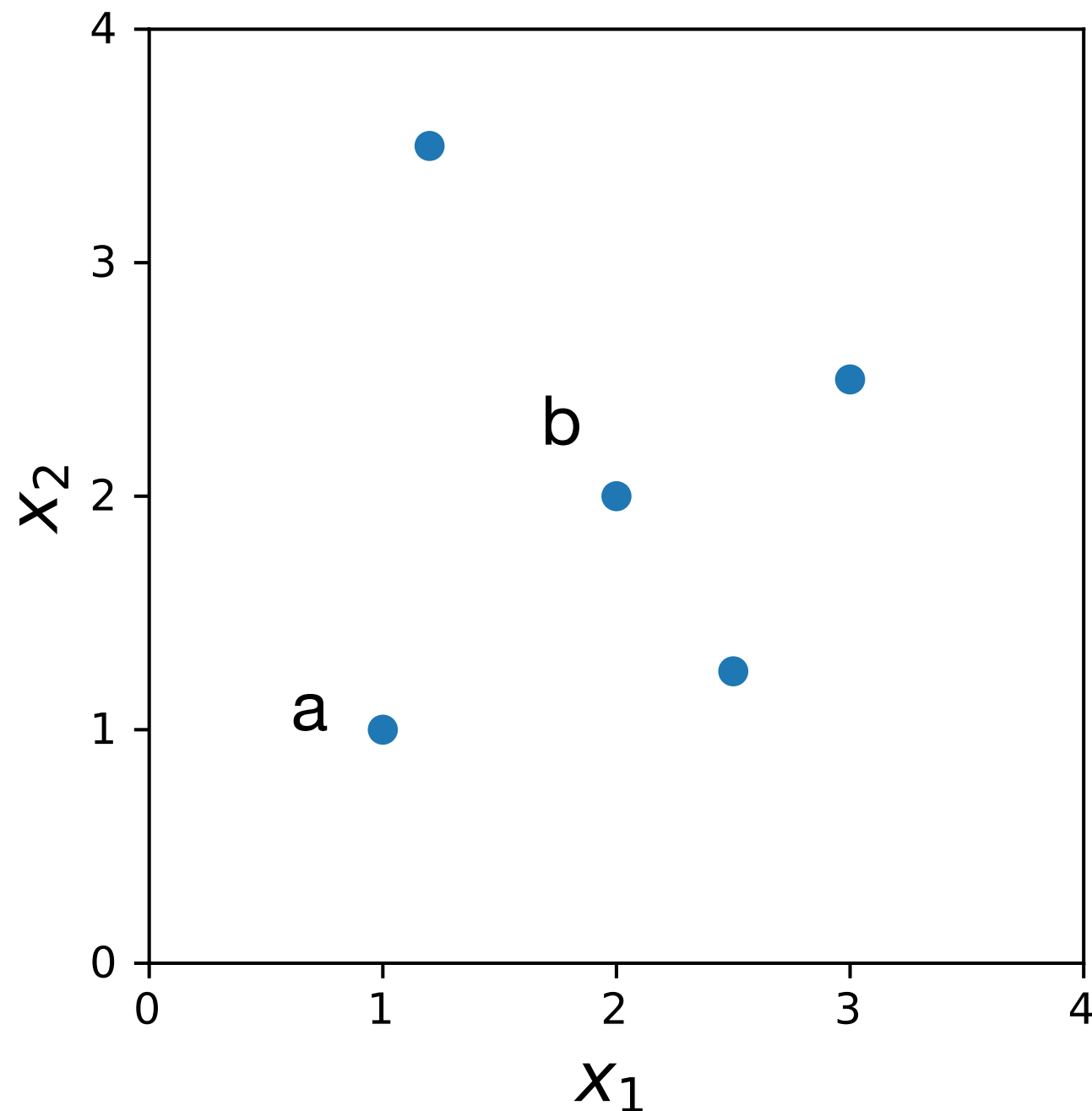
closest_point is the label of $\langle \mathbf{x}^{[i]}, f(\mathbf{x}^{[i]}) \rangle$

Commonly used: Euclidean Distance (L^2)

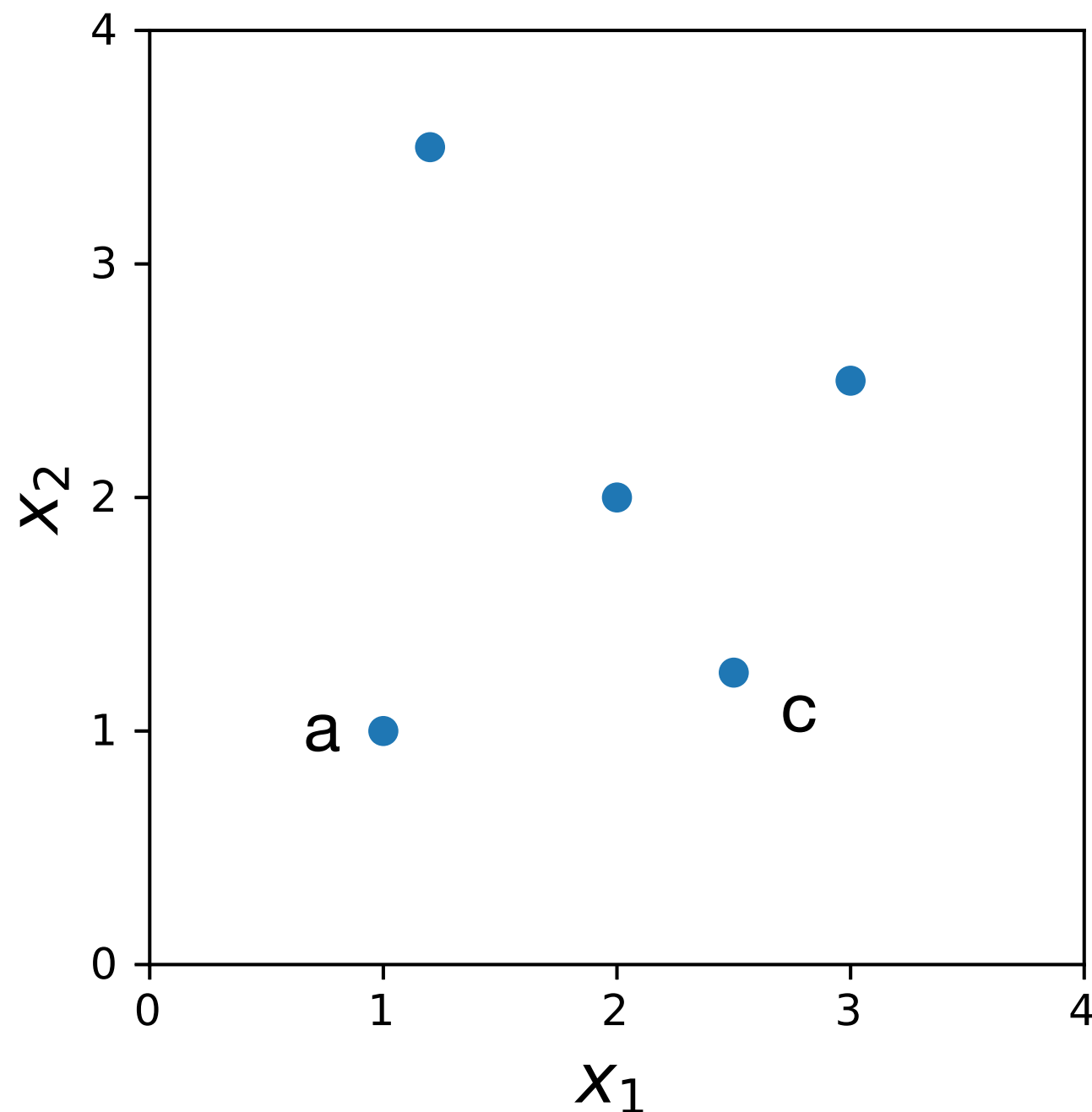
$$d(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \sqrt{\sum_{j=1}^m \left(x_j^{[a]} - x_j^{[b]}\right)^2}$$

Nearest Neighbor Decision Boundary

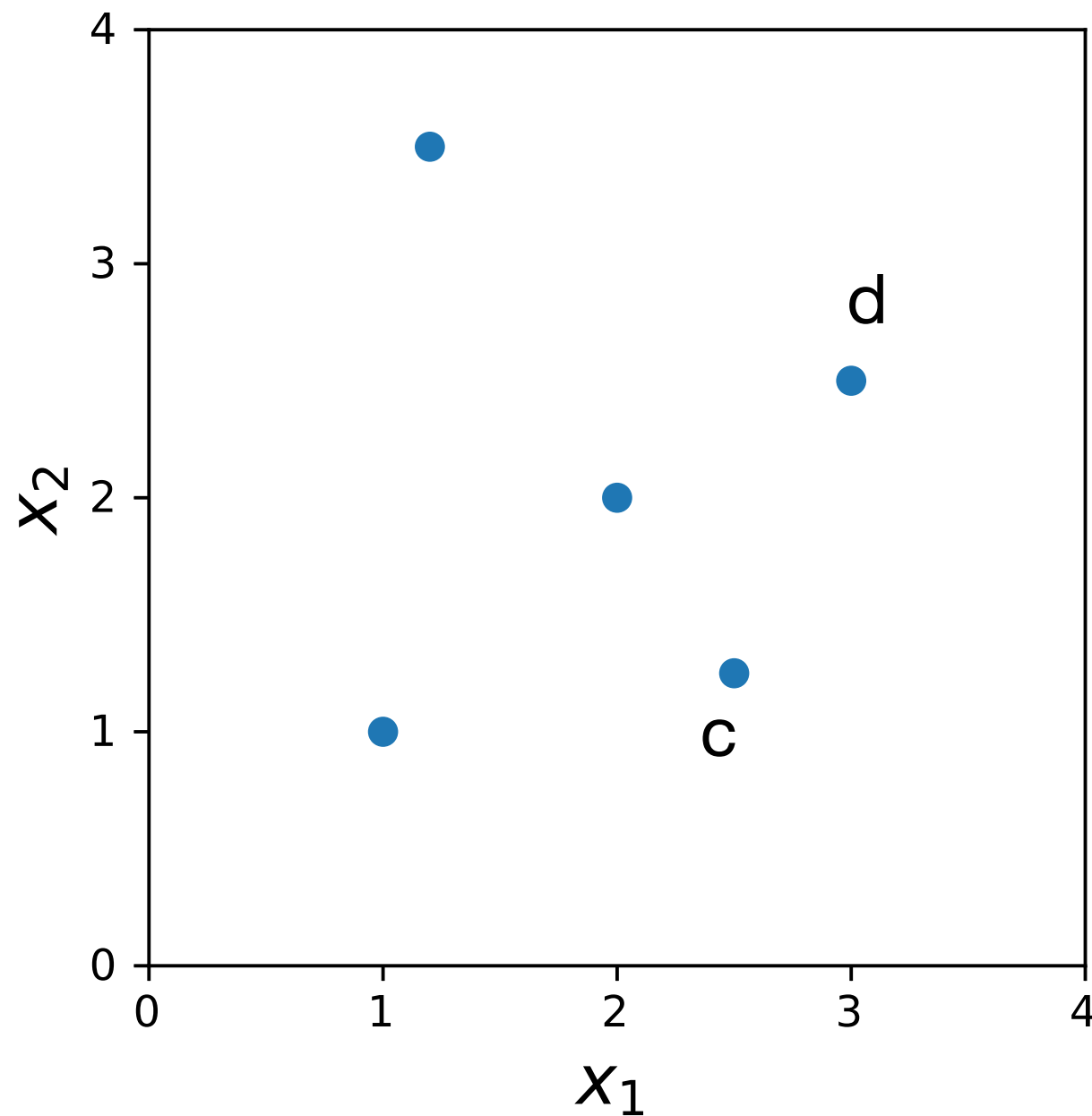
Decision Boundary Between (a) and (b)



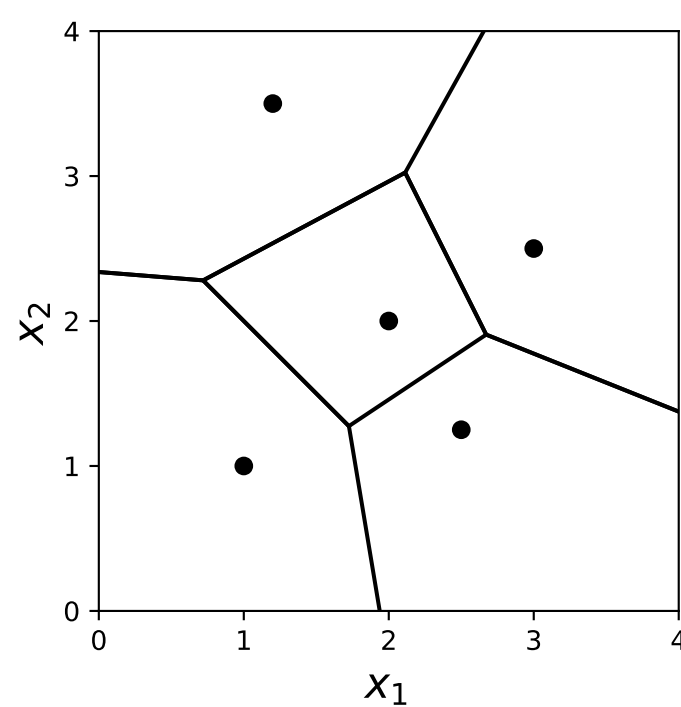
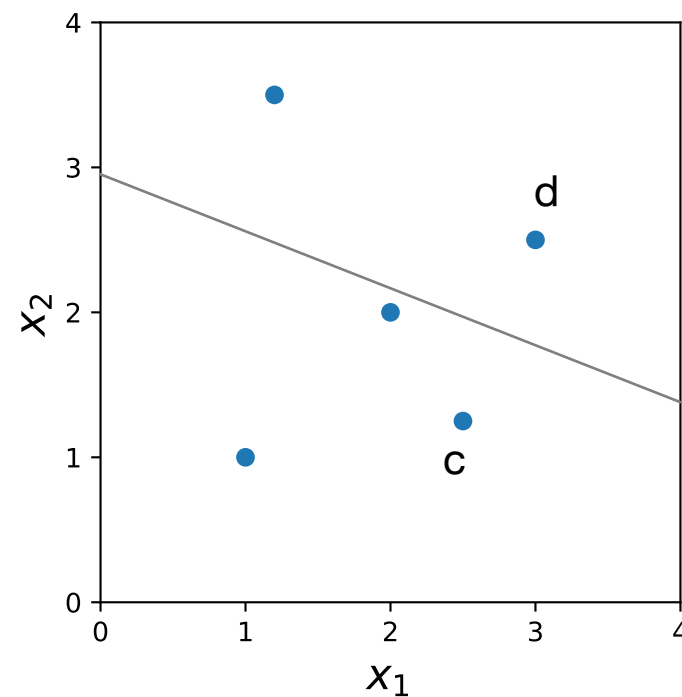
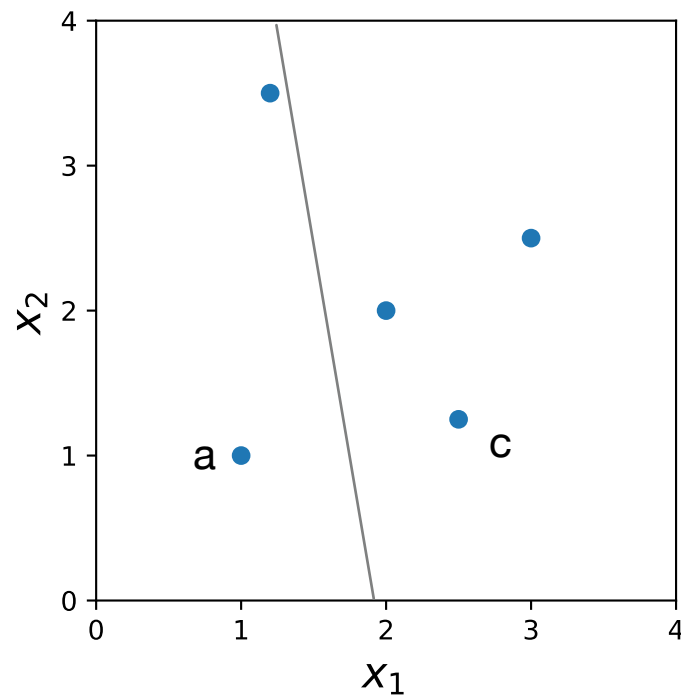
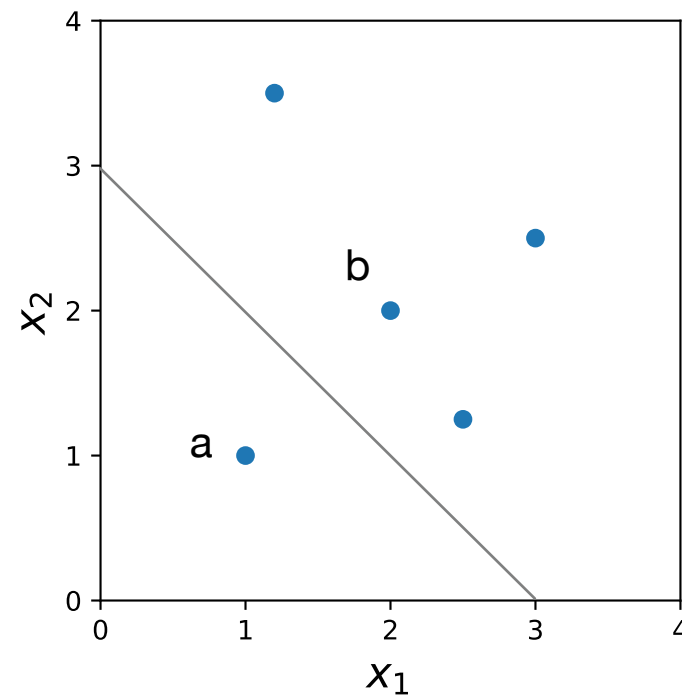
Decision Boundary Between (a) and (c)



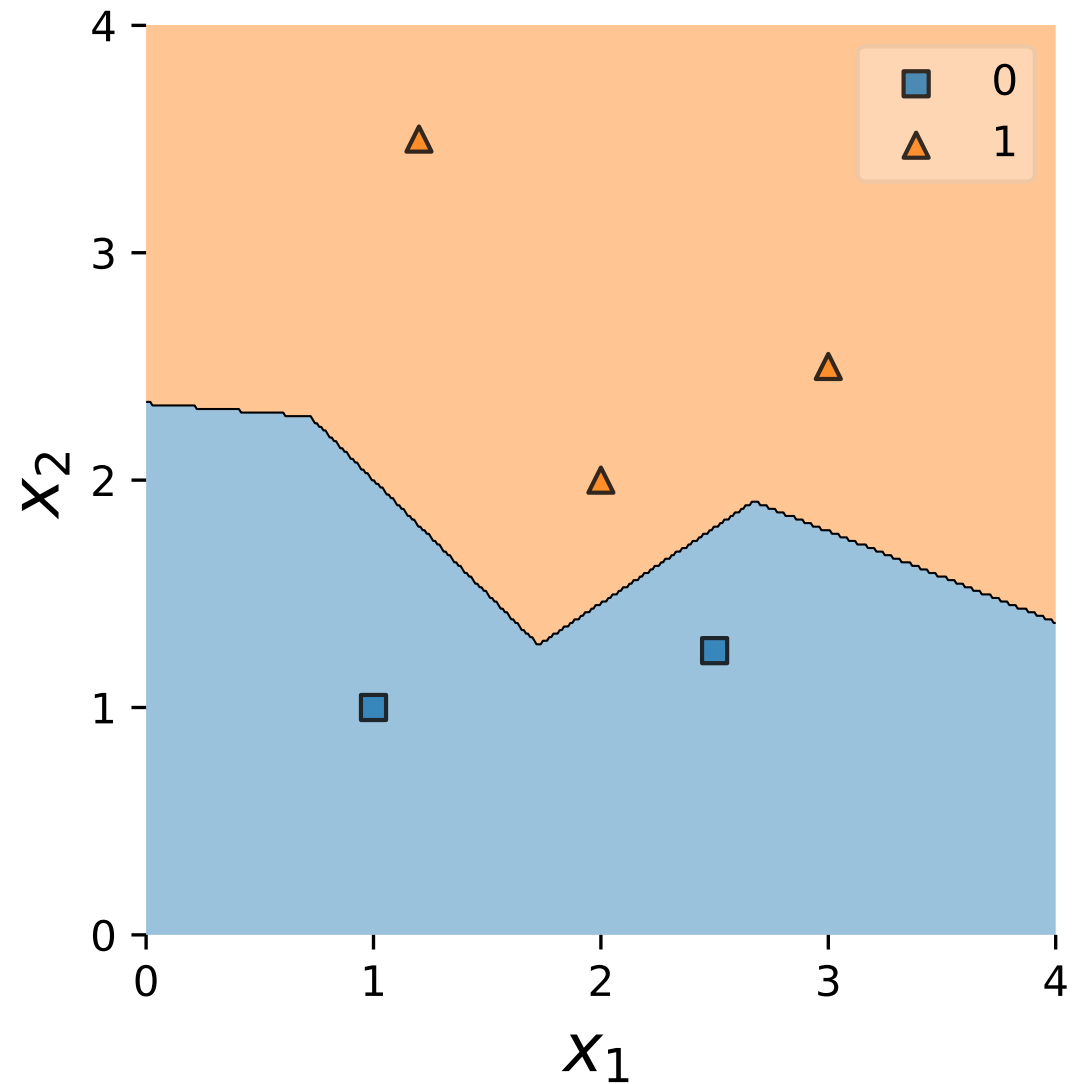
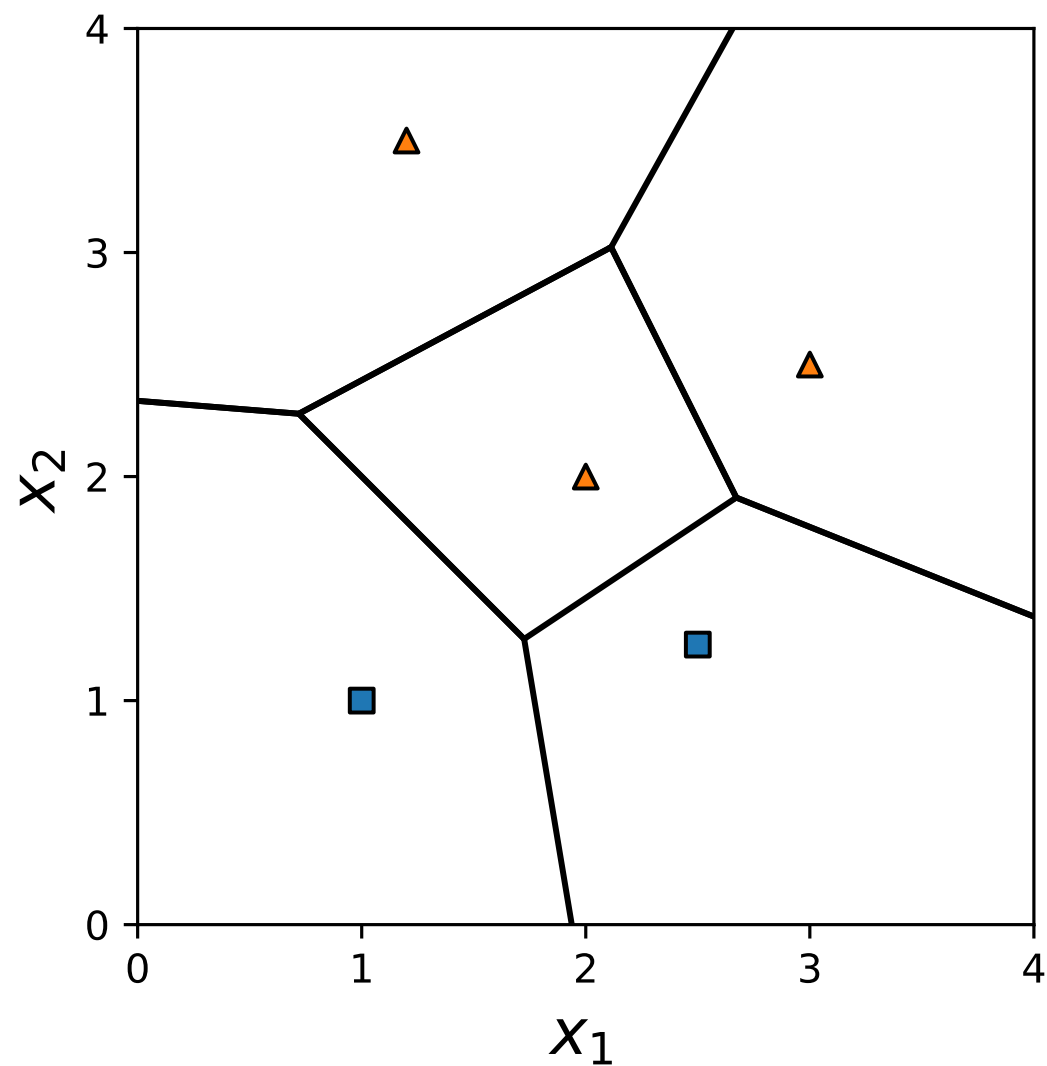
Decision Boundary Between (a) and (c)



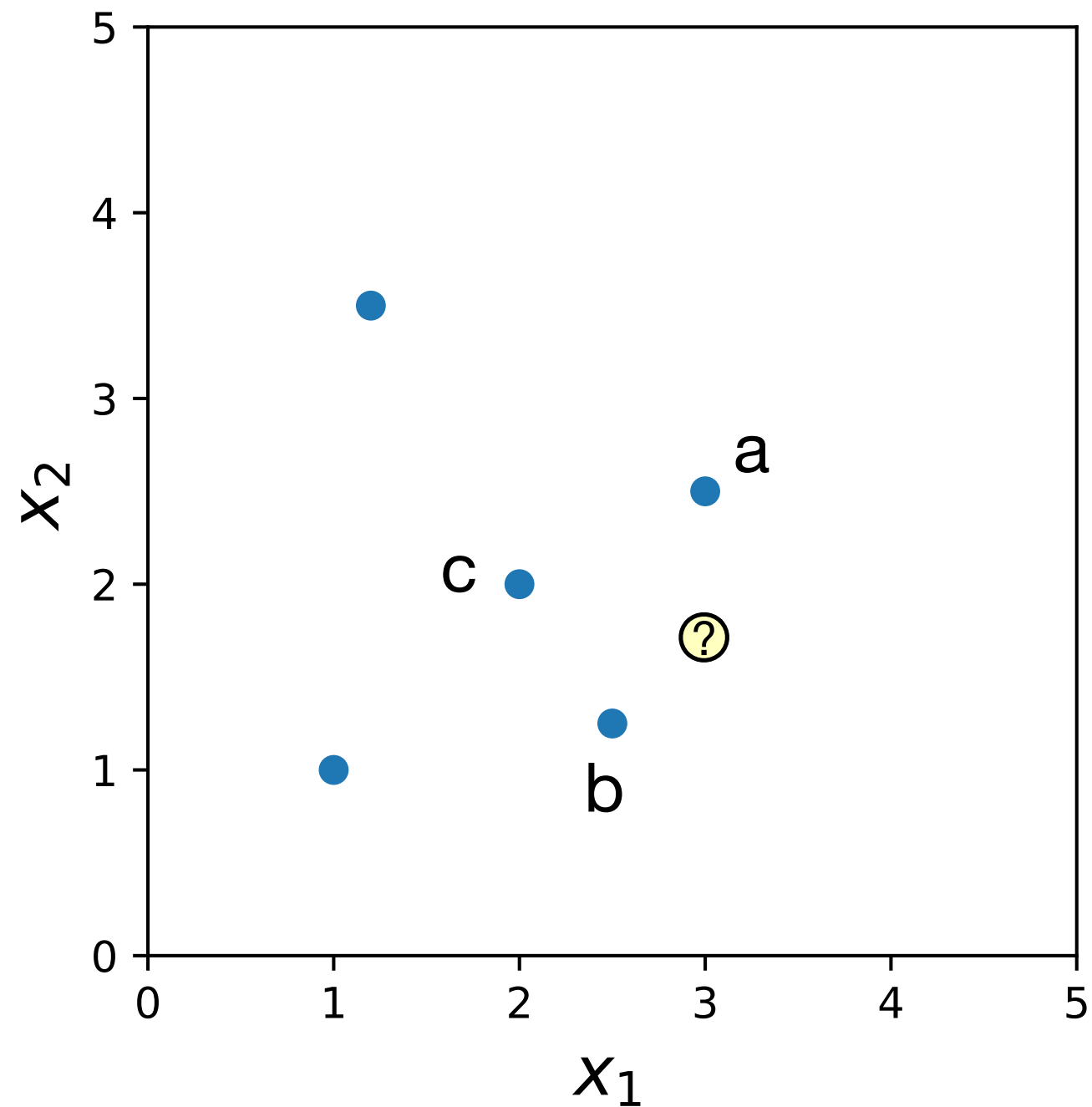
Decision Boundary 1NN



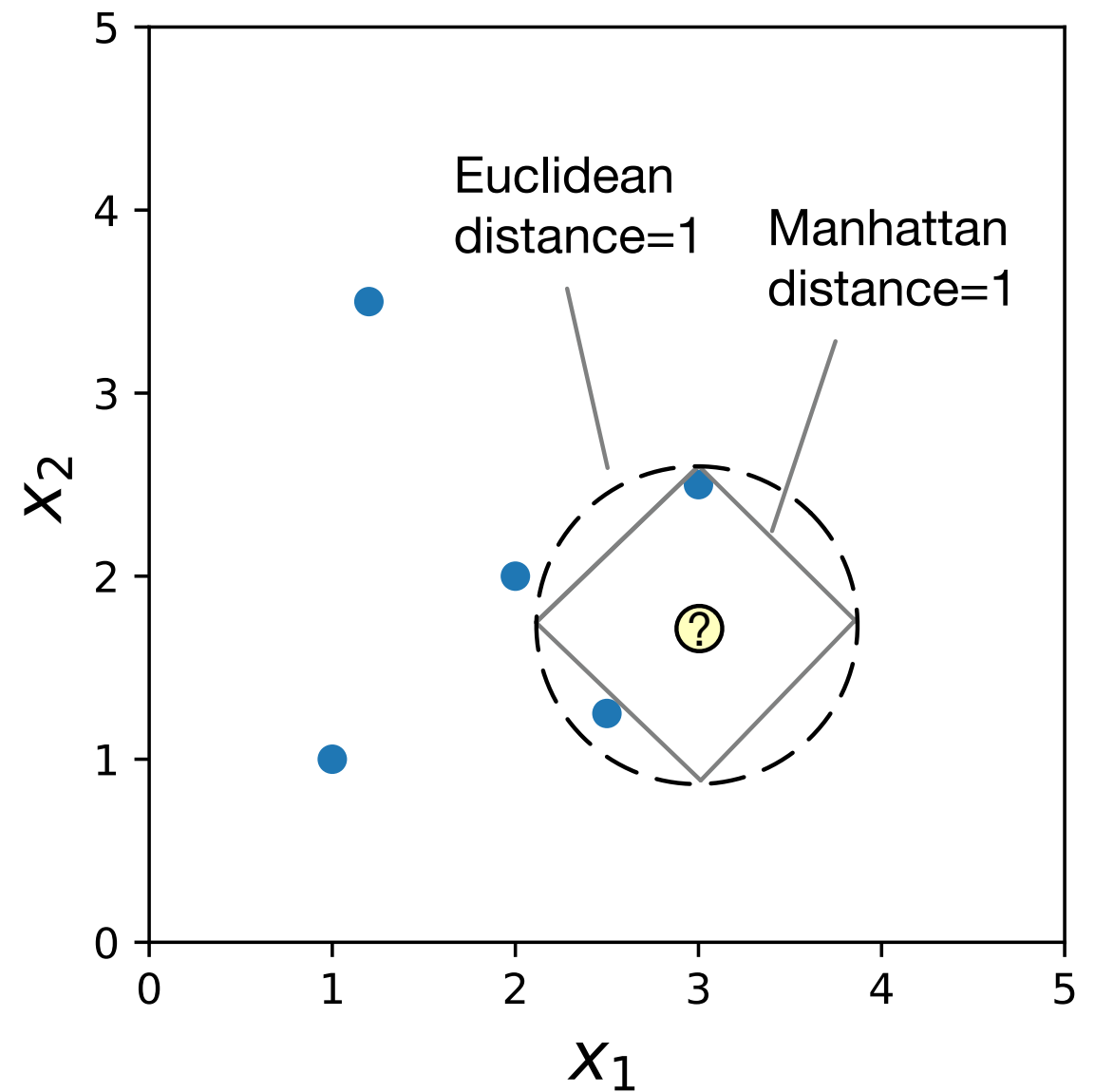
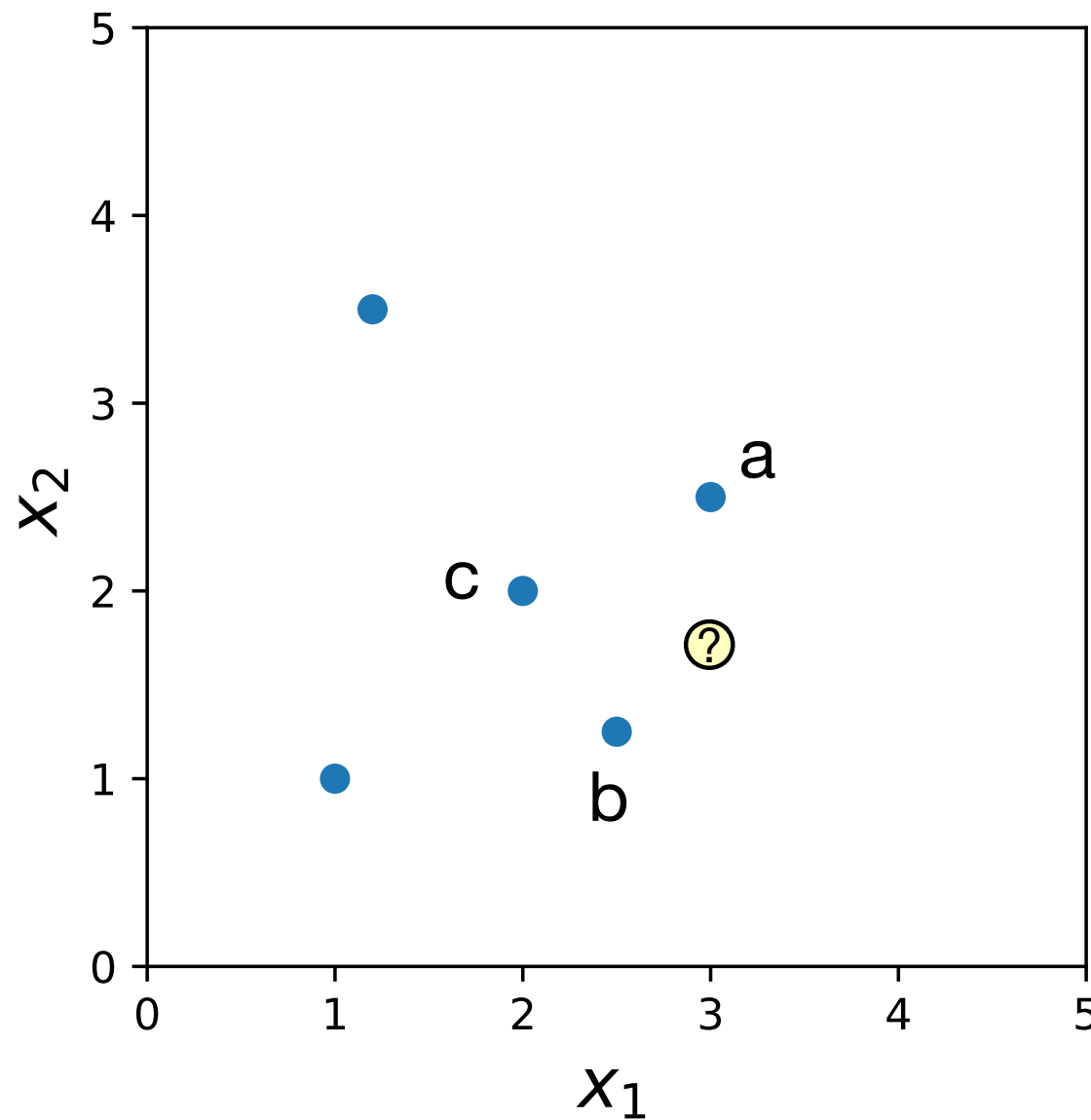
Decision Boundary 1-NN



Which Point is Closest?



Depends on the Distance Measure!



Continuous Distance Measures

Euclidean

Manhattan

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

Mahalanobis

...

Discrete Distance Measures

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

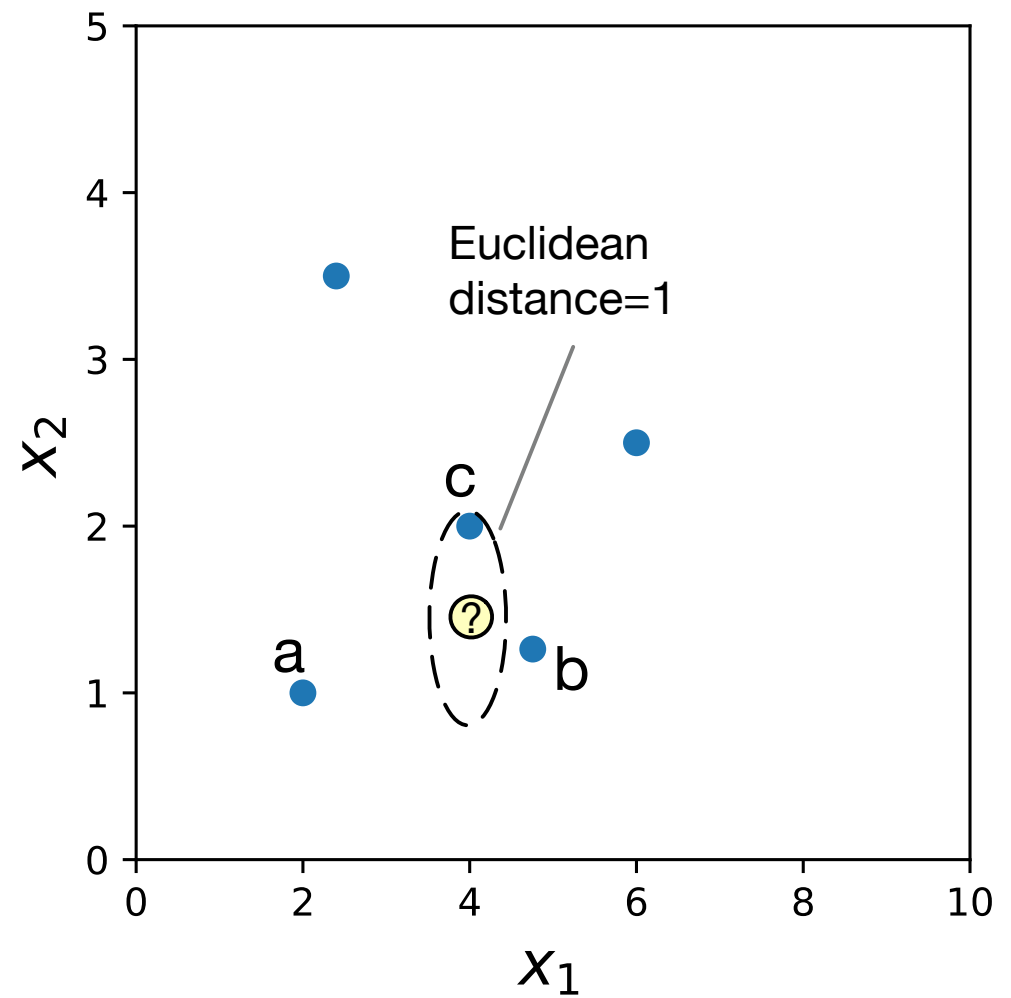
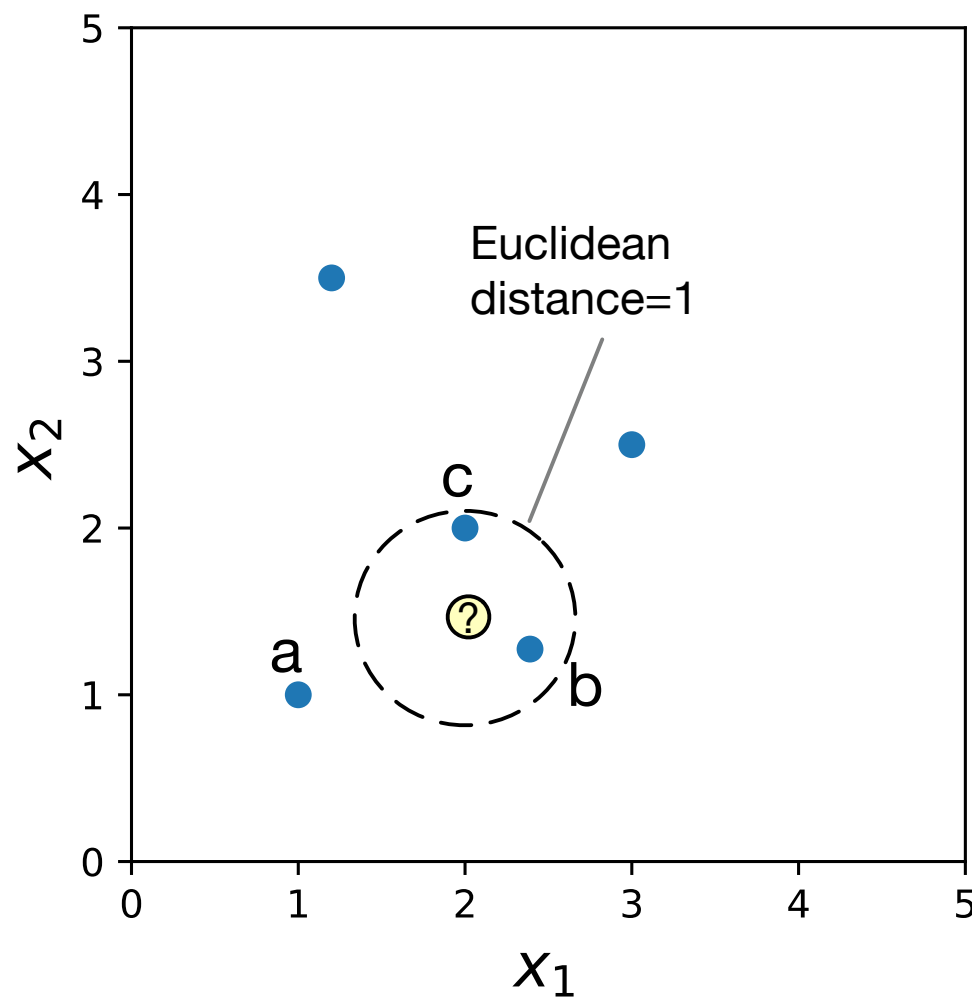
Jaccard/Tanimoto

Cosine similarity

Dice

...

Feature Scaling



A

y:          

Majority vote: 

Purity vote: 

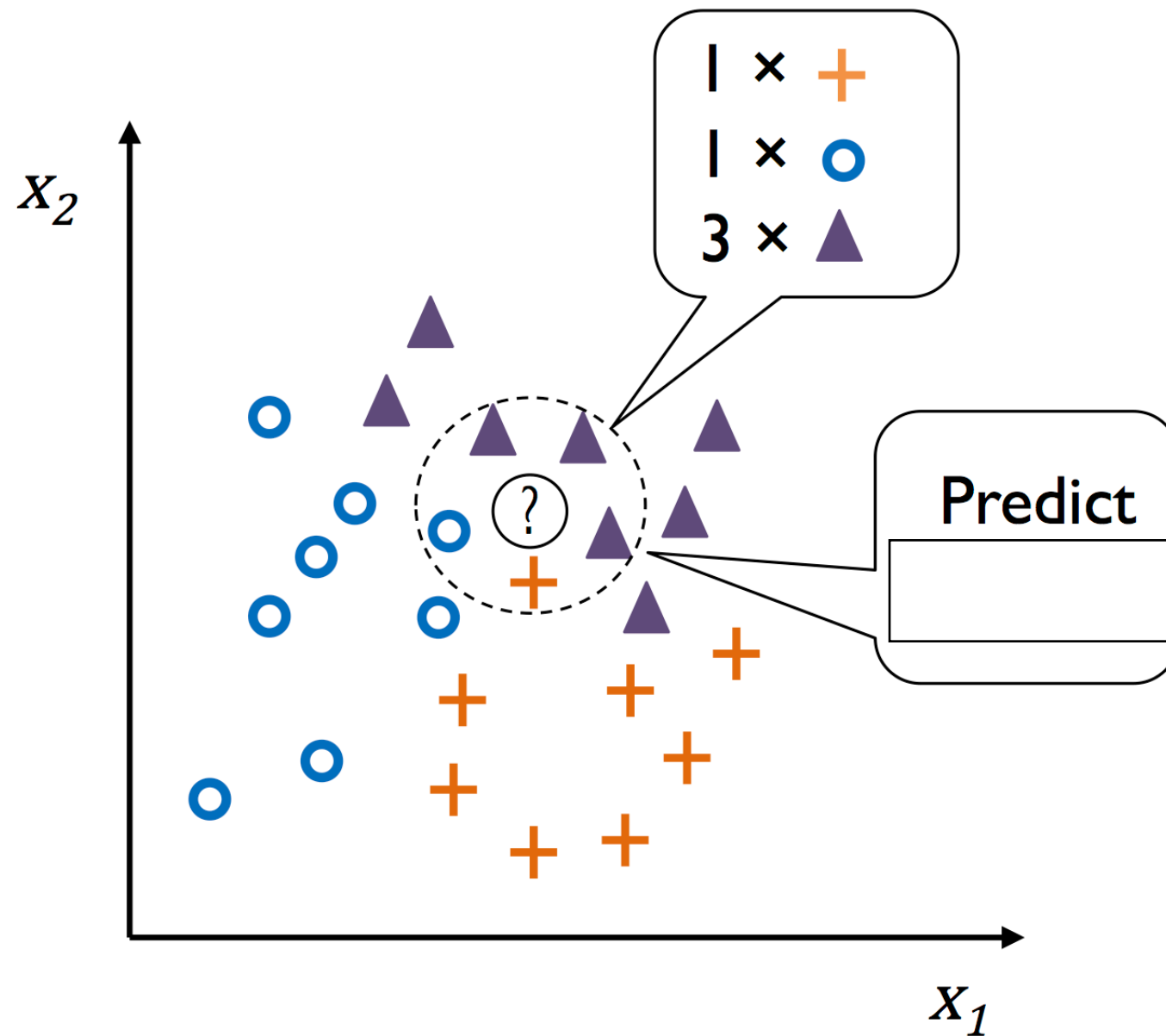
B

y:          

Majority vote: None

Purity vote: 

k -Nearest Neighbors



*k*NN for Classification

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

*k*NN for Classification

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

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

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

k NN for Classification

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

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

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

$$h(\mathbf{x}^{[t]}) = \mathbf{mode}(\{f(\mathbf{x}^{[1]}), \dots, f(\mathbf{x}^{[k]})\})$$

*k*NN for Regression

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

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

Categories (Last Lecture)

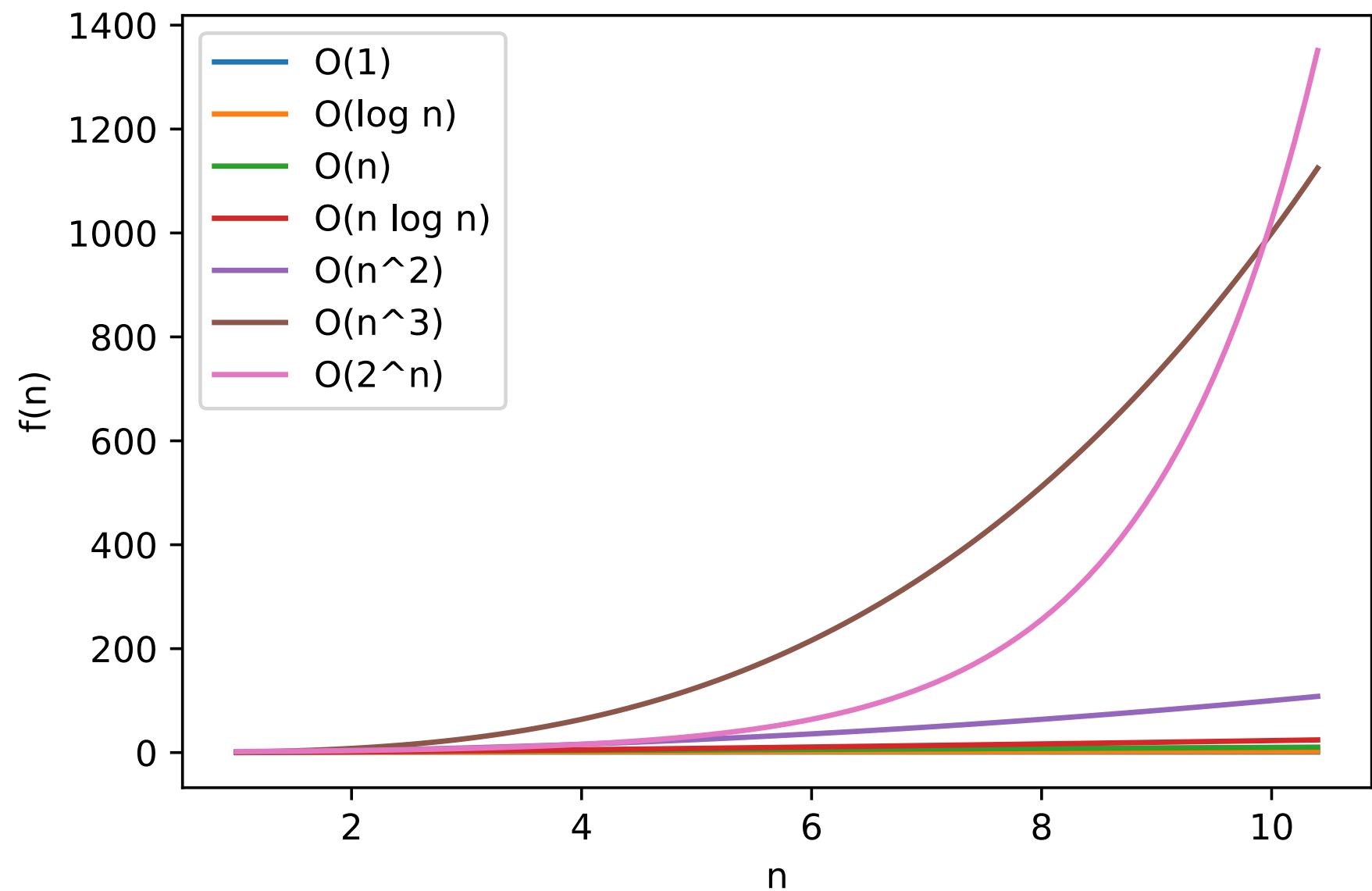
- eager vs lazy;
- batch vs online;
- parametric vs nonparametric;
- discriminative vs generative.

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

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

Distance-weighted k NN

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

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

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>